UNIVERSIDAD DE CIENCIAS Y ARTES DE CHIAPAS

Instituto de Ciencias Básicas y Aplicadas

Centro de Investigación y Desarrollo Tecnológico en Energías Renovables

TESIS

Monitoreo en tiempo real inalámbrico de sistemas energéticos renovables

Que para obtener el grado de:

Maestro en Materiales y Sistemas Energéticos Renovables

Presenta:

Ricardo Marroquín Arreola

Director:

Dr. Jorge Evaristo Conde Díaz

Tuxtla Gutiérrez, Chiapas. **Marzo de 2017**

Tuxtla Gutiérrez, Chiapas
22 de febrero de 2017
Oficio No. DIP-137/17

C. Ricardo Marroquín Arreola
Candidato al Grado de Maestro en
Materiales y Sistemas Energéticos Renovables
P r e s e n t e.

En virtud de que se me ha hecho llegar por escrito la opinión favorable de la Comisión Revisora que analizó su trabajo terminal denominado **"Monitoreo en tiempo real inalámbrico de sistemas energéticos renovables"** y que dicho trabajo cumple con los criterios metodológicos y de contenido, esta Dirección a mi cargo le **autoriza la impresión** del documento mencionado, para la defensa oral del mismo, en el examen que usted sustentará para obtener el Grado de Maestro en Materiales y Sistemas Energéticos Renovables. Se le pide observar las características normativas que debe tener el documento impreso y entregar en esta Dirección un tanto empastado del mismo.

Atentamente

"Por la Cultura de mi Raza"

Dra. María Adelina Schlie Guzmán
Directora.

UNICACH
DIRECCIÓN DE INVESTIGACIÓN
Y POSGRADO

C.c.p. Expediente

Monitoreo en tiempo real inalámbrico de sistemas energéticos renovables

Resumen

En este trabajo se presenta el desarrollo de una aplicación utilizada como centro de monitoreo en tiempo real inalámbrico aplicado a tres diferentes sistemas energéticos renovables utilizando módulos de radiofrecuencia *XBee´s*, los cuales se pueden configurar en una variedad de topologías de redes de comunicación en la frecuencia de los 2.4 GHz. Se logró tener un alcance de cobertura de 120 m a campo abierto entre cada sistema energético seleccionado y el centro de monitoreo con dichos módulos. Los tres sistemas monitoreados son una central fotovoltaica de 2.5 kW, un aerogenerador de 900 W y un deshidratador híbrido solar-eólico.

Para obtener los datos se instalaron sensores de voltaje, corriente, temperatura, humedad, de potencia consumida, sistemas mínimos como arduinos y placas de circuitos impresos realizadas específicamente para el proyecto de acuerdo a las necesidades de cada sistema. Además del monitoreo en tiempo real de manera inalámbrica, los datos obtenidos de cada sistema se almacenaron en una base de datos programada en mysql workbench en la cual se guarda la fecha, hora y los parámetros específicos obtenidos de cada uno de los sistemas y al mismo tiempo la información es enviada a una página web creada en adobe dreamweaver para poder monitorear los sistemas desde cualquier equipo de cómputo que tenga acceso a internet sin la necesidad de estar físicamente en el centro de monitoreo.

Los resultados obtenidos son un monitoreo constante, una base de datos y una página web donde los datos de los sistemas monitoreados son visualizados en tiempo real.

Abstract

This work presents the development of an application used as a wireless real-time **monitoring center** applied to three different renewable energy systems using radio frequency *XBee's* modules, which can be configured in a variety of topologies of communication networks on 2.4 GHz frequency, it was possible to have a coverage range of 120 m open field between each selected energy system and the monitoring center with these modules. The three monitored systems are 2.5 kW photovoltaic plant, 900 W wind turbine and solar-wind hybrid dryer.

To obtain the data from the systems it were installed voltage sensors, current, temperature, humidity, power consumption, electronic boards (arduinos) and circuit boards made specifically for the project according to the needs of each system installed. Besides real-time monitoring wirelessly, the data from each system are stored in a **database** made in mysql workbench in which the date, the time and the obtained specific parameters of each system are saved and simultaneously sent to a **web page** created in adobe dreamweaver to monitor systems from any computer that has access to internet without the need to be physically in the monitoring center.

The results obtained are constant monitoring, database and a web page where the data of monitored systems are displayed in real time.

Agradecimientos

A mi director de tesis el Dr. Jorge Evaristo Conde Díaz por compartirme sus conocimientos, experiencias, su tiempo y guiarme en este trabajo de tesis. Aprecio mucho sus críticas constructivas desde el inicio hasta el final de este trabajo.

Al CONACyT por el financiamiento brindado en este trabajo de investigación.

Al CIDTER por el lugar y excelentes docentes que me brindó.

Al COCyTECH por el apoyo económico otorgado.

Dedicatorias

En especial quiero dedicar este trabajo:

A mis padres Ricardo Marroquín Cruz y Ofelia Arreola Pérez por su apoyo incondicional en todos los momentos de mi vida y su influencia del ejemplo en el trabajo y la sencillez.

A mi esposa Rocío del Carmen y a mi hija Amaia Victoria por brindarme la confianza, alegría, escucharme y apoyarme en los momentos hermosos y difíciles de mi vida.

A mis hermanos Juan, Claudia Patricia y Norma Luz^{+}*.*

A mis amigos Alison, Erick, Luis e Irving por compartirme sus conocimientos.

A Dios todopoderoso por guiarme por la senda del éxito.

Índice

Glosario

Wi-Fi Wireless fidelity

AM amplitud modulada

FM frecuencia modulada

WPAN red inalámbrica de área personal

Kbps Kilobits por segundo

GHz gigahercios

MHz megahercios

IEEE Instituto de ingeniería eléctrica y electrónica

RF radiofrecuencia

USD Dóllar de Estados Unidos

mA miliamperes

mW miliwatts

m metros

V_{DC} volts en corriente directa

kW kilowatt

PC computadora personal

CO_2 Dióxido de carbono

Hz hercios

E energía

c velocidad de la luz

h constante de Planck

kHz kilohercios

P fuente de poder

r radio

I intensidad

P_{rx} potencia de salida

G ganancia

λ longitud

P_{tx} potencia de entrada

s función que depende de una variable

t variable independiente (tiempo)

T constante positiva

W watt

A ampere

A_{DC} amperes en corriente directa

V_{AC} volts en corriente alterna

GND ground (tierra)

mV milivolts

PWM modulación por ancho de pulsos

Kb kilobyte

COM puerto de comunicaciones en serie

.NET dominio de primer nivel derivado de la palabra “NETWORK”

TX transmisor

RX receptor

AC corriente alterna

CSV delimitado por comas

Lista de figuras

Lista de tablas

Introducción

La comunicación inalámbrica o sin cables es aquella en la que la comunicación (emisor/receptor) no se encuentra unida por un medio de propagación físico, sino que utiliza la modulación de ondas electromagnéticas a través del espacio, en este sentido, los dispositivos físicos solo están presentes en los emisores y receptores de la señal, entre los cuales se pueden encontrar antenas, equipos de cómputo, teléfonos móviles, entre otros [1].

En diferentes lugares del mundo se han realizado proyectos con la finalidad de monitorear y controlar de manera inalámbrica un sistema en específico para poder analizar el comportamiento y las diferentes variaciones que se presentan a lo largo de un determinado periodo. Además de ello, comercialmente ya se vende sistemas de monitoreo para equipos como lo son centrales fotovoltaicas, aerogeneradores y secadores, estos sistemas se venden por separado y cada empresa maneja sus diferentes Software dependiendo de los equipos que venden pero en ningún caso manejan un sistema integral de monitoreo [2].

La teoría sobre las comunicaciones electrónicas comenzó a mediados del siglo XIX con el físico inglés James Clerk Maxwell, cuyas investigaciones indicaron que la electricidad y la luz viajan en forma de ondas electromagnéticas y por lo tanto, se relacionan entre sí. El primer sistema de comunicaciones electrónicas fue desarrollado en 1837 por Samuel Morse, quien utilizando la inducción electromagnética pudo transmitir información en forma de puntos, guiones y espacios por medio de un cable metálico.

En 1894, Guglielmo Marconi logró las primeras comunicaciones electrónicas inalámbricas cuando transmitió señales de radio a tres cuartos de milla por la atmósfera de la tierra atravesando la propiedad de su padre, y en 1899 envió el primer mensaje inalámbrico por el canal de la Mancha de Francia a Dover, Inglaterra. Las primeras estaciones de radio Amplitud Modulada (AM) comenzaron a funcionar en 1920 y para 1936 inició la emisión comercial de las señales Frecuencia Modulada [3] (FM).

Planteamiento del problema

Una desventaja de los sistemas de comunicación y transferencia de datos de manera alámbrica es que entre mayor sea la distancia de recorrido, la fidelidad e intensidad de la señal disminuye, por lo que se necesita instalar repetidores o amplificadores a cada determinadas distancias para garantizar que la señal llegue correctamente a su destino, esto conlleva al aumento de los costos de instalación y cableado de los equipos. También dependiendo de la orografía de la región donde se encuentren instalados los sistemas energéticos renovables a monitorear se vuelve muy complicado el acceso a dichas zonas, además en esos lugares por lo general predominan altas temperaturas en el ambiente [2].

Actualmente en los diferentes centros de investigación en energías renovables no existe un sistema de medición, monitoreo, visualización y almacenamiento de datos en tiempo real de los distintos parámetros fundamentales (temperatura, humedad, voltaje, corriente, potencia generada, energía consumida, velocidad del viento, flujo, radiación solar, entre otros), así como sus variaciones en tiempo real que influyen en el funcionamiento y eficiencia de los distintos sistemas energéticos renovables.

Justificación

La creación de un sistema de monitoreo inalámbrico en tiempo real es necesaria para poder visualizar y tener un control estadístico de las variaciones de los parámetros importantes tales como temperatura, humedad, voltaje, corriente, potencia generada y energía consumida en los distintos sistemas energéticos renovables con la finalidad de poder evaluar el desempeño y la eficiencia de cada uno de dichos sistemas tanto en tiempo real como en el transcurso de un determinado periodo, así como tener una base de datos de las variaciones de los parámetros importantes de cada sistema.

Los programas de monitoreo que comercializan las empresas actualmente no permiten la adición de nuevos módulos para poder ser analizados, sino que solamente tienen una interfaz con un número determinado de variables a leer, las cuales ya no se pueden modificar. Además de ello los costos de dichos sistemas de monitoreo son muy elevados, es por ello que el sistema presentado ofrece ventajas notables con respecto a lo que se vende en el mercado ya que además de tener un costo muy económico se pueden añadir más módulos para tener un sistema más complejo.

Objetivos

General

Elaborar un sistema central de monitoreo vía comunicación inalámbrica para su uso como herramienta de visualización, almacenamiento y validación de desempeño de sistemas energéticos renovables.

Particulares

- Hacer un estudio del estado actual de los tres diferentes sistemas energéticos renovables instalados en el CIDTER para determinar el hardware a utilizar en cada uno de ellos.

- Diseñar la interface gráfica para el centro de monitoreo necesaria para la adquisición y visualización de los datos adquiridos por cada uno de los sensores utilizados.

- Integrar y acondicionar las señales en cada uno de los módulos para el envío de datos por internet.

Alcances del proyecto

De acuerdo a los objetivos planteados en este trabajo de investigación, los alcances del proyecto son un monitoreo inalámbrico constante de tres diferentes sistemas energéticos renovables (Aerogenerador de 900 W, Central fotovoltaica autónoma de 2.5 kW y Deshidratador híbrido solar-eólico), la adquisición, visualización y almacenamiento en una base de datos de los siguientes parámetros: fecha, hora, voltaje de entrada, corriente de entrada y energía consumida (para el Aerogenerador y la Central fotovoltaica autónoma); fecha, hora, temperatura de la cámara de secado, temperatura del agua del tanque de almacenamiento y humedad de la cámara de secado (para el Deshidratador Híbrido). Además de esto, los datos son visualizados y graficados simultáneamente en una página web diseñada exclusivamente para el centro de monitoreo.

Estructura de la tesis

El capítulo 1 trata de la tecnología *ZigBee* como medio de comunicación inalámbrica, así como el Software utilizado como herramienta de trabajo, además de los trabajos relacionados con este proyecto.

El capítulo 2 habla del espectro electromagnético, tipos de señales, propagación de ondas de radio, antenas y sistemas de comunicación, los cuales son el fundamento del funcionamiento de las tecnologías de comunicación inalámbrica.

En el capítulo 3 se detalla el proceso que se llevó a cabo para la realización del proyecto utilizando los dispositivos y programas seleccionados como herramientas para la integración de los diferentes sistemas y el centro de monitoreo.

En el capítulo 4 se describen los resultados obtenidos en la realización del proyecto.

CAPÍTULO 1. Sistemas de monitoreo inalámbrico

En la actualidad se cuenta con herramientas básicas de trasmisión y recepción inalámbrica para llevar a cabo algún tipo de investigación o para realizar proyectos en los que la transmisión cableada no nos permite obtener los resultados que queremos. Por este motivo se requiere implementar un dispositivo capaz de realizar una conexión adecuada donde la interferencia electromagnética o cualquier otro medio exterior o interior que pueda ocasionar la perdida de información no afecte directamente al dispositivo, y de esta manera obtener la información con la menor perdida de datos posible. Hay diversas aplicaciones en las que podemos utilizar un enlace inalámbrico como una alternativa para adquirir señales de voltaje o para establecer una comunicación directa con el ordenador, monitorear alguna variable ya sea de temperatura, humedad, potencia, velocidad de viento, entre otras.

1.1 Tecnologías

La comunicación inalámbrica o sin cables es aquella en la que la comunicación (emisor/receptor) no se encuentra unida por un medio de propagación física, sino que utiliza la modulación de ondas electromagnéticas a través del espacio, en este sentido, los dispositivos físicos solo están presentes en los emisores y receptores de la señal, entre los cuales se pueden encontrar antenas, equipos de cómputo, teléfonos móviles, etc.

El monitoreo es el proceso sistemático de recolectar, analizar y utilizar información para hacer un seguimiento al progreso de un programa o actividad en busca del cumplimiento de sus objetivos. Existen diferentes formas de realizar el monitoreo, una de ellas es de manera remota o comúnmente llamada inalámbrica.

La tecnología inalámbrica está cada vez más presente en la vida cotidiana, con tecnologías como *Bluetooth*, *Wi-Fi* y ahora *Zigbee*. Todas estas tecnologías operan en la banda de frecuencia de 2.4 GHz. La ventaja de *Zigbee* ante sus competidores es que al ser un estándar abierto, es innecesario el pago de patentes, lo que lo hace tener un costo bajo con respecto al *Wi-Fi* y el *Bluetooth*, el rango de alcance es mayor, tiene un bajo consumo energético, es altamente confiable, es compatible con distintos fabricantes y tiene una gran capacidad al soportar más de 65,000 nodos independientes.

1.1.1 ZigBee

ZigBee [4] es un estándar abierto para radiocomunicaciones de baja potencia basado en la especificación 802.15.4 [5] de redes inalámbricas de área personal (*WPAN*) [6] por sus siglas en inglés. Incluye un robusto y confiable protocolo de red, de bajo consumo y costo, con servicios de seguridad y capa de aplicación que garantiza la interoperabilidad entre dispositivos. *ZigBee* trabaja a una velocidad de datos de 250 Kbps sobre la banda libre de 2.4 GHz, que está reservado para dispositivos sin licencia en la mayor parte del mundo [7], aunque también soporta las bandas de 868 y 900 MHz.

El estándar *ZigBee* es publicado y administrado por un grupo de empresas y fabricantes denominado *ZigBee Alliance* [8]. Por lo tanto el término *ZigBee* es en realidad una marca registrada de este grupo y no un estándar técnico. Sin embargo, para fines no comerciales, la especificación *ZigBee* se encuentra disponible de forma abierta y libre al público. De esta forma la relación entre *IEEE* 802.15.4 [5] y *ZigBee Alliance* es similar a aquella entre *IEEE* 802.11 [9] y *Wi-Fi Alliance* [10].

Una de las características más destacables de *ZigBee* es su soporte a redes de mallas o *mesh networking* [11]. En una red de malla los nodos se interconectan entre sí de forma que existen múltiples rutas posibles entre ellos. Las conexiones entre nodos se actualizan dinámicamente y se optimizan mediante sofisticadas tablas de encaminamiento integradas.

Las redes de mallas o *mesh ZigBee*, son descentralizadas y cada nodo es capaz de descubrir la red por sí mismo. En otras palabras, *ZigBee* es una tecnología de red de comunicaciones inalámbricas, que permite que sus nodos se comuniquen entre sí en forma independiente, es decir, los nodos pueden no mandar directamente sus datos al concentrador, sino que pueden transmitirlos a otros nodos de la red, para que lleguen a su destino. Esto es vital en el caso de que uno de los nodos falle, donde la comunicación se re-arma en forma automática, por intermedio de otros nodos [12], es por ello que este tipo de configuración es necesaria en todo sistema de monitoreo y comunicación inalámbrica que conste de muchos dispositivos a monitorear tal como es el caso del presente proyecto.

Para que se pueda trabajar con una red *mesh ZigBee* se necesita de nodos que forman parte de la misma, los cuales se clasifican en 3 tipos.

1. **Coordinador:** es el nodo más importante de la red, puesto que es el encargado de crearla. Por este motivo, sólo puede existir un único coordinador por red. Además, el coordinador se encarga de asignar direcciones a los nodos que se van uniendo a la red, mantiene la seguridad, administra el sistema y actúa de enlace con otras redes u otros dispositivos.

2. ***Router*:** es un nodo *ZigBee* plenamente operativo [13]. Puede unirse a redes existentes, transmitir, recibir y enrutar información. Actúa como intermediario entre nodos que no pueden comunicarse directamente entre sí, ampliando el alcance efectivo de la red. Además puede permitir a otros nodos el ingreso en la red. Su consumo es mayor que el de los terminales puesto que, al tener responsabilidades de enrutamiento, no puede desconectarse en ningún momento.

3. **Terminal:** es un nodo provisto con la funcionalidad básica para unirse a la red y enviar y recibir información. No puede actuar como intermediario entre otros nodos. Necesita que un *router* o el coordinador sea su nodo padre y le permita el acceso a la red. Puede entrar de forma cíclica en un modo de bajo consumo para ahorrar energía [14].

En la figura 1.1 se puede observar cómo se compone una red *ZigBee*, en la cual participan forzosamente un coordinador, elemento de mayor importancia ya que es el encargado de crear la red, dos o más *routers*, quienes transmiten, reciben y guían la información, por último, las terminales de la red, las cuales intercambian datos con los *routers*.

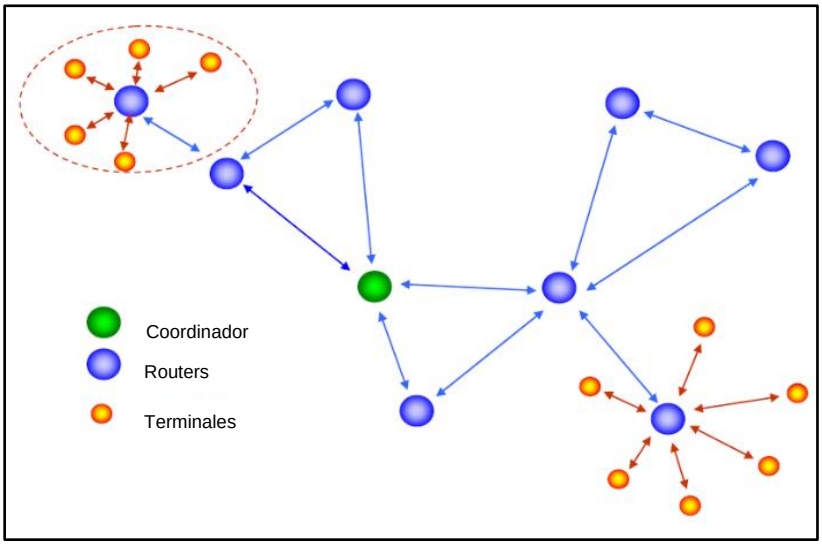

Figura 1.1. *Ejemplo de topología típica de una red ZigBee [14].*

Principales características de *ZigBee*

- **Bajo costo:** Estándar abierto que hace innecesario el pago de patentes.
- **Bajo consumo energético:** Permite prolongar ampliamente la vida de las baterías.
- **Alta confiabilidad:** Redes malladas con redundancia de enlaces y canales y posibilidad de enrutamiento alternativo automático.
- **Despliegue de red sencillo:** Fácil de montar y administrar gracias a las redes *ad-hoc* y al enrutamiento automático.
- **Compatibilidad:** Al tratarse de un estándar abierto se garantiza la interoperabilidad entre dispositivos de diferentes fabricantes.
- **Gran capacidad:** Una misma red puede soportar más de 65,000 nodos independientes.

Beneficios de una red *Mesh ZigBee*

- Es posible llevar los mensajes de un nodo a otro por diferentes caminos.
- Cada nodo tiene sus propias comunicaciones con todos los demás.
- Cada nodo añadido extiende el alcance de las comunicaciones.
- Es más segura, si un nodo falla, el otro se hará cargo del tráfico.
- Elimina gastos y problemas de cableado [15].

Todo lo anterior se deriva de una comparativa entre las principales tecnologías de comunicación inalámbrica, es decir, *ZigBee*, *Bluetooth* y *Wi-Fi*. En la Tabla I [16] se puede observar que *ZigBee* es la tecnología más económica en costo, tiene menor consumo de corriente, una mayor capacidad de nodos y un rango de alcance superior en comparación de las otras 2 tecnologías, es por ello que se eligió una red *ZigBee*, aunque esta tenga una menor velocidad de transmisión RF que las demás, esto no afecta al proyecto ya que el tipo de datos que se van a transmitir solo son señales de radiofrecuencia y no archivos de audio ni video.

Tabla I. *Comparación de tecnologías de comunicación inalámbrica [16].*

Características	***ZigBee* 802.15.4**	***Bluetooth* 802.15.1**	***Wi-Fi* 802.11**
Costo del Chip (USD)	1	3	4
Consumo de corriente (mA)	30	65-170	350
Capacidad de red (nodos)	65000	30	7
Ancho de banda (MHz)	5	1	25
Velocidad de transmisión RF (Kbps)	250	1,000-3,000	54,000
Potencia de transmisión (mW)	1-2	1-100	40-200
Frecuencia de radio (GHz)	0.868; 0.915; 2.4	2.4	2.4
Rango de trabajo (m)	1-1,100	1-50	30-100

1.1.2 XBee

XBee [17] es el hardware *ZigBee* que se trata de una gama de módulos de radio fabricados por *Digi International* [18] con una amplia variedad de modelos, componentes, firmware, potencias de transmisión y antenas. Trabajan con un voltaje de operación de 3.3 a 5 V_{DC} y, aunque incorporan hasta 20 pines para distintas operaciones como reinicio o modo de bajo consumo, pueden funcionar haciendo uso tan solo de sus 4 pines principales: alimentación, tierra, entrada y salida de datos, en la figura 1.2 se puede observar un módulo *XBee* serie 2.

Estos dispositivos cuentan con un microchip que, además de ocuparse de las tareas propias de la comunicación de radio, les permite realizar operaciones con un nivel de lógica muy básico, como por ejemplo, leer directamente datos de sensores y retransmitirlos sin tener que contar con procesado externo, para lógica más

avanzada deben conectarse a otro dispositivo que se encargue de ello, como un microcontrolador más complejo.

Figura 1.2. *Módulo de radiofrecuencia XBee series 2 de Digi International [17].*

Los módulos *XBee* se clasifican principalmente en dos tipos:

- **Serie 1.** Esta serie de módulos proporciona soporte para comunicaciones punto a punto de forma simple y basándose en el estándar 802.15.4. Su principal ventaja es la sencillez en conexiones punto a punto ya que únicamente se configura a uno como el emisor y al otro como el receptor, sin la necesidad de configurar el número de canal ni las direcciones de fuente y destino de los módulos.

- **Serie 2.** Estos módulos permiten implementar redes malladas siguiendo el protocolo *ZigBee*, además poseen un mayor alcance y un menor consumo de corriente que la serie 1. Estas características convierten a dichos módulos en los idóneos para implementar sistemas de comunicación inalámbrica con topología de red en malla (ver Tabla II).

Cabe mencionar que los módulos *XBee* de la serie 1 y la serie 2 son incompatibles entre sí, y no pueden interactuar entre ellos de ninguna forma. Esto debido a que los módulos de la serie 1 no tienen las opciones de configuración de canal, direcciones de fuente y destino como en el caso de los de la serie 2.

Tabla II. *Comparación entre la serie 1 y la serie 2 de módulos de radio XBee[17].*

	Series 1	**Series 2**
Rango de alcance con obstáculos	30 m	40 m
Rango de alcance en línea recta y sin obstáculos	100 m	120 m
Consumo de corriente de transmisión/recepción	45/50 mA	40/40 mA
Firmware	802.15.4 punto a punto	ZB ZigBee mesh
Pines de entrada y salida digitales	8	11
Pines de entrada analógica	7	4
Topología de árbol y malla	NO	SI

Los radios *XBee* presentan dos modos de funcionamiento, denominados modos *AT* y *API.*

- **Modo AT:** Se trata de un modo transparente donde la información recibida por el pin de entrada de datos es retransmitida por radio sin ninguna modificación. Los datos pueden ser enviados a un único destinatario (punto a punto) o bien pueden ser transmitidos a múltiples destinos (*broadcast*).
- **Modo API:** En este modo, los datos son encapsulados dentro de una trama que proporciona varias funcionalidades adicionales, tales como direccionamiento, confirmación de paquetes y *checksum*, entre otras. Existen varios tipos de tramas API y cada una ofrece distintas prestaciones: comandos de configuración, transmisión de datos, reconocimientos de mensajes, etcétera.

Para programar los módulos *XBee* se debe cargar el firmware deseado y modificar el valor de los distintos parámetros que determinan su comportamiento. Para ello se utiliza un software distribuido por *Digi* [18], denominado XCTU [19] y que se puede descargar de forma gratuita desde su página web. Este programa esta solo disponible para entornos Windows, permite actualizar el firmware del módulo, determinar su comportamiento como coordinador, router o terminal, leer la configuración actual, realizar pruebas de alcance o facilitar el acceso a las

funcionalidades del modo API, entre otras posibilidades. Basta con conectar el módulo a un ordenador haciendo uso de un adaptador USB y usar el software para programarlo con la configuración deseada [19].

A pesar de que la única forma de actualizar el firmware de un módulo *XBee* es mediante el software XCTU, para modificar otras configuraciones se puede hacer uso de cualquier otro programa terminal serie, entrar en modo comando y reconfigurar el módulo mediante el envío de comandos AT [20].

1.1.3 Arduino

Arduino [41] es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos. Arduino puede tomar información del entorno de toda una gama de sensores a través de sus pines de entrada y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El micro-controlador en la placa Arduino se programa mediante el lenguaje de programación Arduino [21] (basado en *Wiring* [22]) y el entorno de desarrollo Arduino. Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software.

El hardware consiste en una placa con un micro-controlador Atmel AVR y varios puertos de entradas/salidas, tanto digitales como analógicos, así como salidas PWM y de comunicaciones para el control de objetos físicos (Leds, servos, botones, etc.). Los micro-controladores más usados son el ATmega328 y el ATmega168 para las placas básicas, el ATmega1280 para la de mayor capacidad y el ATmega8 para las placas más antiguas. Todos estos micro-controladores incluyen un cargador de arranque (bootloader) que permite programar al Arduino a través del puerto serial sin necesidad de utilizar un programador externo, ya que durante el arranque de Arduino el bootloader comprueba si se está intentando programar el Arduino. Con ello se garantiza una manera más simple para empezar a trabajar [22].

Existe una amplia gama de placas prefabricadas que se ajustan a las distintas necesidades de cada proyecto, ofreciendo distintos tamaños, potencias, costos, modelos de micro-controlador, necesidades de alimentación, número de entradas/salidas analógicas y digitales y otras prestaciones específicas. Algunos de los modelos oficiales más utilizados son Uno, Leonardo, Mega ADK, Pro, Mini, Nano y Fio [42], en la figura 1.3 se puede observar el Arduino Mega ADK.

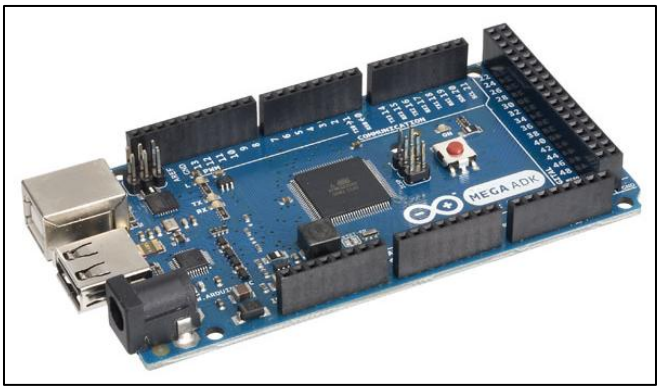

Figura 1.3. *Placa de Arduino Mega ADK [41].*

1.1.4 Microsoft Visual C# 2010 Express

Microsoft Visual C# 2010 Express [23] es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en *.NET Framework*, C# es simple, eficaz, con seguridad de tipos y orientado a objetos.

Visual C# es una implementación del lenguaje C# de Microsoft. *Visual Studio* ofrece compatibilidad con *Visual C#* con un completo editor de código, un compilador, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y de fácil uso y otras herramientas. La biblioteca de clases de *.NET Framework* ofrece acceso a numerosos servicios de sistema operativo y a otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa.

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada plataforma .NET, y a los servicios antes comentados se les denomina servicios Web [23].

1.2 Situación actual del monitoreo de sistemas

Actualmente en los distintos centros de investigación y desarrollo en energías renovables se tienen instalados y funcionando diferentes prototipos y sistemas, algunos más automatizados que otros, buscando alternativas de producción de energía útil proveniente de fuentes de energías que no dependan de los fósiles y que a la vez no generen un daño al medio ambiente. La mayoría de esos sistemas no cuentan con el adecuado monitoreo que se necesita para poder llevar un seguimiento del funcionamiento y la eficiencia de dichos sistemas [24].

En el caso específico del Centro de Investigación y Desarrollo Tecnológico en Energías Renovables (CIDTER) de la Universidad de Ciencias y Artes de Chiapas (UNICACH) actualmente se cuenta con diferentes sistemas energéticos renovables instalados y funcionando, pero aún no se cuenta con un monitoreo individual o un centro de monitoreo en tiempo real en el cual se concentren todos estos sistemas para poder evaluar la eficiencia de cada uno de ellos y conocer en realidad la situación actual de los equipos para poder tomar medidas de mejoramiento de los sistemas.

Dentro de los sistemas que están instalados se encuentra un Deshidratador híbrido solar-eólico, el cual se utiliza para deshidratar frutas y hojas. Durante el día utiliza la energía térmico-solar para calentar la cámara de secado y por la noche utiliza un sistema alterno que consta de un tanque con una resistencia para calentar el agua

y hacerla recircular en la cámara de secado para poder seguir deshidratando durante la noche o en días nublados o lluviosos, este sistema alterno obtiene la energía mediante un Aerogenerador.

Otro de los sistemas instalados es una central fotovoltaica autónoma de 2.5 kW, la cual proporciona energía al Centro de Desarrollo y Evaluación de Biodigestores del CIDTER mediante un regulador de carga, un banco de baterías y un inversor. También se cuenta con un Aerogenerador de 900 W que se utiliza para alimentar un sistema aislado, dicho aerogenerador también cuenta con su regulador de carga, un banco de baterías y un inversor.

De todos los sistemas instalados actualmente solo se van a monitorear 3, los cuales son un Aerogenerador de 900 W, una Central fotovoltaica autónoma de 2.5 kW y deshidratador híbrido solar-eólico.

1.3 Trabajos relacionados

Actualmente debido al constante avance en las tecnologías de comunicación y sistemas de producción de energía cada vez más eficientes y complejos se busca tener un control de dichos sistemas, así como saber las variaciones que se puedan presentar. Por lo que en diferentes lugares del mundo se han realizado proyectos con la finalidad de monitorear y controlar de manera inalámbrica un sistema en específico para poder analizar el comportamiento y las diferentes variaciones que se presentan a lo largo de un determinado periodo. Además de ello, comercialmente ya se vende sistemas de monitoreo para equipos como lo son centrales fotovoltaicas, aerogeneradores y secadores, estos sistemas se venden por separado y cada empresa maneja sus programas dependiendo de los equipos que venden pero en ningún caso manejan un sistema integral de monitoreo [11].

En 2011 se publicó un artículo acerca de un sistema de monitoreo y control de energía en tiempo real basado en una red de sensores *Zigbee*. El trabajo fue realizado por Woong Hee Kim, Sunyoung Lee y Jongwoon Hwang, de la Universidad de Saarlandes, Alemania. En dicho trabajo se propuso un sistema de gestión de

energía basado en redes de sensores inalámbricos, el cual se compone de dos componentes principales: una red de sensores inalámbricos y un portal de acceso inteligente. El sistema permite a los usuarios ahorrar energía mediante la supervisión y el control de los aparatos electrodomésticos a través de la web y dispositivos móviles [11]. En un equipo de cómputo se conectó un dispositivo *Zigbee* mediante una interfaz USB, que sirve como puerta de entrada de los datos al equipo de cómputo, la aplicación es independiente del sistema operativo ya que fue realizada en Java. Los datos entrantes de cada sensor contienen el estado actual del dispositivo y el consumo de energía, estos datos son almacenados en una base de datos para más tarde utilizar una interfaz web o aplicación móvil.

Como conclusión se tiene que con la retroalimentación efectiva sobre el consumo de energía y el control de los aparatos domésticos, los usuarios pueden estar motivados y animados a cambiar su comportamiento en el uso de energía, como apagar las luces o reducir el calor, estos pequeños cambios en el comportamiento pueden conducir a importantes ahorros en el consumo de la energía [11].

En la Universidad del Norte de Texas, Estados Unidos se publicó un artículo acerca del diseño de un sistema inalámbrico de sistemas utilizando Raspberry PI y Arduino para aplicaciones de monitoreo ambiental en el año de 2014, los autores son Sheikh Ferdoush y Xinrong Li. En dicho trabajo se describe un sistema de red de sensores inalámbricos que se desarrolló a través de plataformas de hardware de código abierto, Arduino y Raspberry Pi [6].

Se desplegó un sistema que consta de una estación base, 3 nodos de router, y 3 sensores nodos. Los módulos *XBee* en nodos de *router* y de los sensores están configurados con el dispositivo enrutador y el módulo *XBee* en la estación de base está configurado con el tipo de dispositivo coordinador. Como resultado, los módulos *XBee* pueden formar en una topología de red en malla utilizando los protocolos de red *ZigBee*. El dispositivo coordinador puede apoyar un máximo de 10 nodos secundarios, y un dispositivo enrutador puede soportar un máximo de 12 nodos secundarios.

En la figura 1.4 se puede observar una gráfica en internet de la aplicación realizada para el monitoreo ambiental donde se observa la humedad relacionada a un nodo de monitoreo [6].

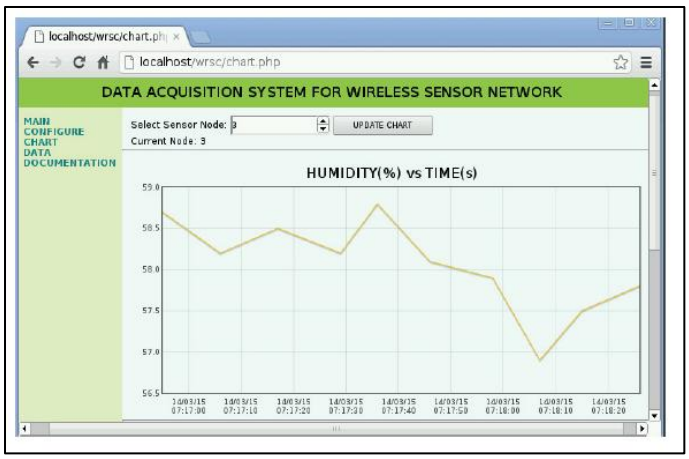

Figura 1.4. *Interfaz web del sistema de sensores inalámbrico [6].*

En 2015 se desarrolló un sistema de adquisición de datos utilizando una red de sensores, *ZigBee* y *Bluetooth*, los autores son Sushabhan Choudhury, Piyush Kuchhal, Rajesh Singh y Anita, todos de la Universidad de Estudios de Energía y Petróleo, Odisha, India [25]. El trabajo aborda la propuesta de un sistema de adquisición de datos para las fábricas y la industria dedicada a la vigilancia del medio ambiente, los parámetros monitoreados son la temperatura, humedad, el nivel de los gases presentes en la atmósfera, el movimiento de cualquier persona cerca de las zonas restringidas y a la vez transmitir estos datos a la sala de control de forma inalámbrica, así como a la persona afectada y esta a su vez, a través de un Smartphone, puede dar orden a la sala de control para que se actúe.

Los datos se obtuvieron de los nodos sensores al nodo sala de control utilizando la red *ZigBee* y luego se retransmitieron los datos deseados a teléfonos inteligentes, pestañas y ordenadores que utilizan la red *Bluetooth*. El nodo sensor contiene sensores de salida analógica como la temperatura, el gas y los sensores de salida digital como el sonido y el metal. La pantalla LCD se utiliza para visualizar los parámetros del sensor [25].

El módem de RF se utiliza para transmitir los valores al nodo sala de control utilizando topologías de red en estrella y malla. El nodo de sala de control contiene módulo transceptor *ZigBee* para recibir la información y el módem *Bluetooth* para poner a disposición los datos deseados para los teléfonos inteligentes, pestañas y PC. La red *ZigBee* y la *Bluetooth* funcionan a 9600 baudios y 2,4 GHz de frecuencia en la banda ISM. Los nodos *ZigBee* son capaces de transmitir la información hasta 100 metros y para la comunicación de larga distancia, se utiliza multi-hopping y el módem de *Bluetooth* transmite la información a 9 metros de distancia.

Telmo Moya Javier y Goglino Daniel Hoyos, ambos del Instituto Nacional de Energías No Convencionales de Argentina, desarrollaron en 2010 un proyecto de red de sensores y control inalámbrico para un sistema de generación de vapor solar térmico [26]. En dicho trabajo se presenta una red de sensores y actuadores diseñada para controlar sistemas complejos como un generador de vapor utilizando concentradores fresnel y células de medición o control, según las necesidades del sistema, las cuales se comunican entre sí mediante una red *ZigBee* y el protocolo modbus. Para esto se desarrolló una microcomputadora que se puede conectar a una red *ZigBee* a través de un módulo *XBee* además de ello se probó una red de microcontroladores conectada a un sistema de cómputo utilizando *ZigBee* y *Modbus* en una comunicación punto a punto para lograr una eficiente y sencilla comunicación de datos entre cada uno de los equipos, ver figura 1.5.

Figura 1.5. *Instalación del fresnel a monitorear en el proyecto [26].*

En la Universidad del Caribe, Cancún Quintana Roo, Pedro Canché Martín y Mario Marín Montoya realizaron un Prototipo de red de sensores de temperatura, utilizando módulos *XBee*, para control de incendios [27], el cual consta de 5 nodos:

un nodo maestro o coordinador, y 4 nodos terminales o esclavos, que se comunican inalámbricamente y una computadora conectada a través del puerto serial con el nodo maestro, además se utilizó el software X-CTU como programa interfaz.

Con las pruebas realizadas, se pudo comprobar que el uso de aparatos de enlace inalámbricos o de microondas no afectó el desempeño del prototipo, pues no presentó pérdidas de datos o señal, además de ello, Los resultados obtenidos tienen una mayor eficiencia debido a que pueden ser leídos desde cualquier punto utilizando un explorador de internet, sin la necesidad de estar presente en el lugar donde son generados los datos.

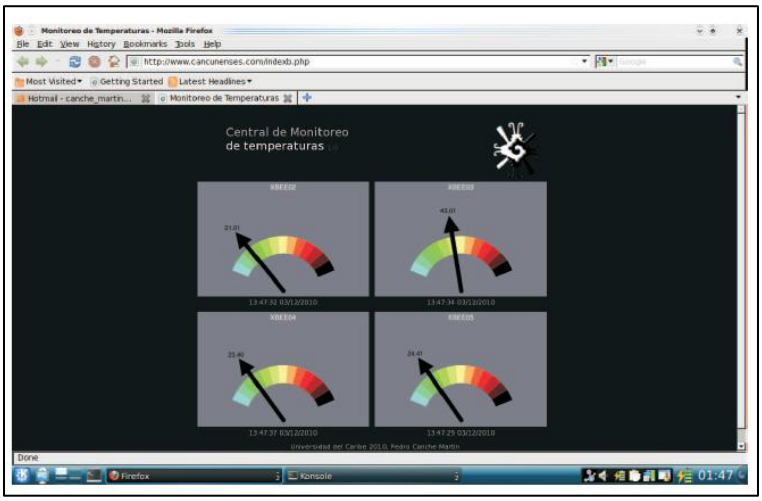

Figura 1.6. *Página web de visualización de los datos adquiridos por el X-CTU [27].*

En el 2013 se realizó un proyecto de desarrollo e implementación utilizando Arduino y *ZigBee* con un sensor ultrasónico para el control de nivel de llenado, el proyecto fue realizado por Juan Jesús Almansa Madrigal de la Universidad de Rovira y Virgili, España [22].

En dicho proyecto se diseñó una placa base, que conectada a un *XBee* y a un sensor ultrasónico constituye un módulo *End Device* que se instaló en el techo del contenedor, dicho *XBee* envía datos a un coordinador situado en una caja de comunicaciones que recibe los datos y los procesa mediante un equipo de cómputo, a la vez que dichos datos se van guardando en una base de datos que registra el número y estado del contenedor.

El proyecto mostró la viabilidad en la utilización de un micro-controlador Atmel de Arduino, un sensor ultrasónico y un sistema de comunicación sin hilos basado en protocolo *ZigBee* para controlar el nivel de llenado en forma remota de unos contenedores situados en la calle como se observa en la figura 1.7.

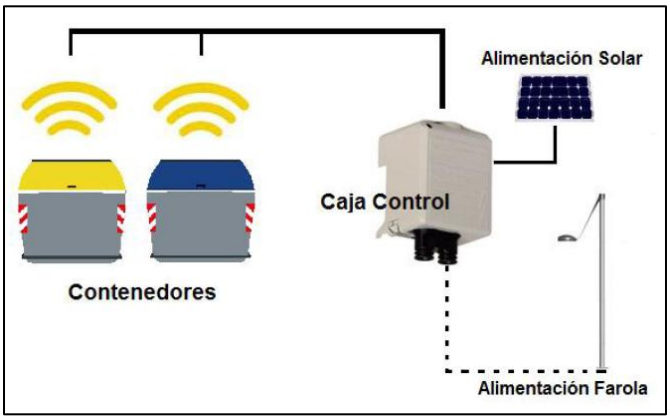

Figura 1.7. *Esquema de comunicación de caja de control con contenedores [22].*

José Salvador Montesinos Navarro, de la Universidad Politécnica de Cartagena realizó un proyecto de red de sensores auto-configurable mediante tecnologías *ZigBee* y Arduino con monitorización por aplicación android en el año de 2013 [4]. En dicho trabajo se aborda el diseño e implementación de una red inalámbrica de sensores de temperatura y luz capaz de reconfigurarse automáticamente al variar la disponibilidad de los nodos que la componen mediante la tecnología *ZigBee*.

Los datos de los sensores, junto con la ruta que siguen los paquetes dentro de la red, son recopilados por un nodo coordinador capaz de transmitir dicha información vía Bluetooth a un dispositivo Android, donde una aplicación se encarga de monitorizar la red, mostrando dicha información tanto en forma gráfica como en texto.

El proyecto logró conseguir el objetivo de diseñar e implementar una red mallada inalámbrica de sensores que son capaces de reconfigurarse automáticamente para adaptarse a condiciones cambiantes tales como desconexión de nodos, modificaciones en su disposición, caídas de enlaces o ingreso de nuevos nodos a la red [4].

1.4 Sistemas de monitoreo en el mercado

El sector más explotado en el campo de las energías renovables es el solar fotovoltaico en cuanto a sistemas de monitoreo se refiere, la mayoría de las empresas que venden paneles solares también ofrecen un sistema de monitoreo para sus productos ya sea de un sistema de 2, 3 paneles hasta centrales fotovoltaicas de grandes capacidades en kW [28].

Krannich Solar [29] es una empresa alemana que ofrece su software de monitoreo Solar-Log para los sistemas de paneles solares que venden, además como un extra, dicho softwaremuestra la radiación solar de la zona así como la contribución al medio ambiente del sistema instalado al no producir CO_2, estos datos también los puede ofrecer en los Smartphones, los precios de sus productos de monitoreo oscilan entre los $30,000.00 y $75,000.00 MN [29].

La empresa Enphase Energy también tiene en el mercado su propio Software llamado Enlighten [30] el cual es capaz de visualizar la potencia generada y consumida, gráficas de historial y consumo de potencia de los paneles, además de ello, ofrece un servicio de diagnóstico remoto y acceso a reporte de fallas en su servidor.

Energía Alternativa de México (ENALMEX) es una empresa Mexicana que ofrece distintos servicios y productos del sector energético renovable, entre ellos se encuentra la unidad de comunicación y monitoreo de datos SMA Sunny Boy [31] la cual se comunica de forma inalámbrica a través de Bluetooth con hasta 12 inversores y muestra de manera gráfica los datos de rendimiento, muestra simultáneamente salida de energía, la producción diaria de energía y la producción total de energía del sistema.

También puede estar configurado para visualizar otros parámetros, tales como la compensación de CO_2 global o valor en dólares de la electricidad producida. La información del sistema puede ser grabada y transferida a través de una interfaz USB o a través de la página web de Sunny Portal mediante el sistema llamado SMA Sunny WebBox para el almacenamiento a largo plazo, la visualización y evaluación.

El Sunny Beam [31] tiene un alcance estándar de hasta 150 metros que se puede ampliar usando un repetidor Bluetooth opcional, sus precios oscilan entre los $25,000.00 y $60,000.00 MN [31].

Otra empresa que se encuentra en esta rama del sector comercial e industrial es Exemys [32] de Argentina que se dedica a la fabricación de productos de conectividad para el monitoreo, adquisición y control industrial que ha desarrollado una radio serie industrial basada en tecnología de redes Mesh ZigBee. Esta línea de productos permite conectar puertos seriales RS232, RS485 o USB en forma totalmente inalámbrica ya sea en conexiones punto a punto o en multipunto, lo cual permite extender a la distancia a un puerto serial, mediante un canal de radio con este tipo de tecnología.

Exemys también ofrece su producto llamado wRemote el cual es un dispositivo electrónico que permite alcanzar distancias de hasta 2 kilómetros entre los dispositivos con las antenas incorporadas de 2 dBi (con línea de vista), o mayores distancias con el uso de antenas externas de 2.4 GHz. El sistema está formado por 2 dispositivos, un concentrador y nodos remotos, todos los dispositivos comparten una misma red de datos denominada red *Mesh*. Este tipo de transmisión de datos que ofrece dicha empresa es a través de la tecnología Wireless Fidelity [33] (Wi-Fi), sus productos de telemetría tienen un costo desde los $10,000.00 hasta los $35,000.00 MN [32].

1.4.1 Comparativa entre sistemas comerciales y el centro de monitoreo

Después de mencionar algunas empresas que compiten actualmente en el mercado del monitoreo y control de producción de electricidad utilizando paneles solares se puede realizar una comparativa de los precios entre los productos ofrecidos por dichas empresas y el sistema de monitoreo desarrollado en este trabajo.

En la Tabla III se enlistan los precios comerciales que manejan las empresas y el precio del sistema de monitoreo a partir de su costo de diseño y construcción, en la cual se puede observar que el precio de dicho sistema de monitoreo es de un

25.58% del costo total del equipo más caro que se menciona en este trabajo (el sistema de monitoreo SIR-R2600-M250-E) ofrecido por la empresa mexicana ENALMEX, generando un ahorro del 74.42%.

Ahora bien, si se compara con el costo del equipo comercial más económico (que lo ofrece la empresa MONSOLAR) se observa que el costo del sistema de monitoreo es el 55.85% del costo total del "*Solar-log 300 Wifi*" ofrecido por dicha empresa, obteniéndose un ahorro del 44.15%. Lo que convierte al sistema de monitoreo en una opción viable al tener un precio muy por debajo del costo comercial de equipos destinados al mismo fin.

Tabla III. *Comparación entre precios comerciales y precio del sistema de monitoreo.*

Empresa	**Producto ofrecido**	**Precio en MXN**
Energía Alternativa de Mexico	SIR-R2600-M250-E	$40,122.00
SMA Solar Technology Ag	Sunny WebBox with Bluetooth	$33,400.00
Telemetic	DABIN-1083-ST-HL	$21,437.00
Monsolar	Solar-log 300 Wifi	$18,382.00
Universidad de Ciencias y Artes de Chiapas	Sistema de monitoreo	$10,267.00

Además del costo, las ventajas que ofrece el sistema de monitoreo son varias, entre las cuales se encuentra la característica de ser un sistema compuesto por módulos individuales de monitoreo, lo cual significa que se pueden monitorear diferentes sistemas energéticos renovables y/o agregarse más sensores para tener un centro de monitoreo más completo, por ejemplo, sensores de carga y descarga de baterías, piranómetros, anemómetros, sensores de temperatura para monitorear el calentamiento de las baterías, sensores de vibración, entre otros dependiendo del sistema al cual se decida agregar más sensores.

En la Tabla IV se describen las ventajas y desventajas del sistema de monitoreo con respecto a los equipos comerciales.

***Tabla IV*.** *Ventajas y desventajas del sistema de monitoreo.*

Ventajas	**Desventajas**
Bajo costo con respecto a equipos comerciales	Requiere de varias aplicaciones abiertas al mismo tiempo para monitorear
Utiliza módulos de monitoreo individuales para cada sistema	Necesita un servidor web para alojar la página web donde se visualizan los sistemas
Se pueden agregar más sensores a cada sistema	Requiere un reinicio de los módulos de monitoreo y del servidor cada determinado periodo
Se pueden agregar más módulos de monitoreo (es decir, más sistemas a monitorear)	
Sistema de monitoreo integral (no solo se monitorean paneles solares, también aerogeneradores y secadores)	
El diseño y la electrónica ocupada es simple con respecto a otros sistemas	
El sistema ocupa equipos comerciales para realizar el monitoreo	
Si se daña un sensor o un equipo, simplemente se reemplaza (no se tiene que reemplazar el sistema de monitoreo completo)	

Como se puede observar en la tabla anterior, el número de ventajas del sistema es mayor con respecto a sus desventajas, lo cual lo hace bastante viable para su uso como herramienta de monitoreo de diferentes sistemas energéticos renovables y se puede aplicar tanto en la investigación, en la industria o en el comercio.

1.5 Conclusiones

- Para el monitoreo remoto de sistemas es más rentable utilizar equipos de radiofrecuencia que hacerlo de manera alámbrica ya que disminuye la complejidad de la comunicación y los costos de adquisición e instalación.
- La tecnología *ZigBee* es ideal para realizar un sistema de monitoreo inalámbrico gracias a que es configurable, tiene un mayor rango de alcance y se pueden realizar topologías en malla, y el *Bluetooth* y *Wi-Fi* no pueden realizar este tipo de configuraciones.
- Los módulos de radiofrecuencia *XBee* son los dispositivos adecuados para realizar sistemas de comunicación con topología de redes en mallas, además de que son muy económicos y de estándar libre.
- Los dispositivos Arduino son una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexible y fácil de usar, es decir, es un estándar libre y altamente configurable.
- El software ideal para aplicaciones de monitoreo en la plataforma de Windows es *Microsoft Visual C# 2010 Express* ya que es de comunicación libre y orientado a objetos, lo que lo convierte en una potente herramienta para la visualización de datos y gráficas en tiempo real.
- Actualmente aún no se ha desarrollado un proyecto de monitoreo inalámbrico de sistemas energéticos renovables que integre diferentes plataformas de evaluación, únicamente se han realizado proyectos de monitoreo para sistemas específicos.
- Los costos de los programas que venden las empresas que se dedican al monitoreo son relativamente costosos y van desde los $25,000.00 hasta los $100,000.00 MN dependiendo de las características de cada sistema.

Referencias

[1] Como funcionan las redes inalámbricas, Preston Gralla, Eric Lindley, Anaya Multimedia, 2006.

[2] Julián Adolfo Ramírez, Jaime Alberto Buitrago, Jorge Iván Marín, Red de sensores de larga distancia usando ZigBee para el monitoreo y la gestión del riesgo en el departamento del Quindío-Colombia, Revista de Investigaciones, 25(2014), 63–72.

[3] Sistemas de comunicaciones electrónicas, Wayne Tomasi, cuarta edición, Prentice Hall, 2003.

[4] José Salvador Montesinos navarro, Red de sensores auto-configurable mediante tecnologías ZigBee y Arduino con monitorización por aplicación Android, Universidad Politécnica de Cartagena, Septiembre de 2013.

[5] Product Manual v1.xAx - 802.15.4 Protocol for OEM RF ModulePart Numbers: XB24-...-001,XBP24-...-001 IEEE 802.15.4 OEM RF Modules byMaxStream.

[6] Sheikh Ferdoush, Xinrong Li, Wireless sensor network system desing using raspberry Pi and Arduino for Enviromental Monitoring Applications, Procedia computer science 34(2014) 103-110.

[7] Fundamentals of Wireless LANs, CISCO Networking Academy Program, 2003.

[8] Farihah Shariff, Nasrudin Abd Rahim, Hew Wooi Ping, Zigbee-based data acquisition system for online monitoring of grid-connected photovoltaic system, Expert systems with applications 42(2015) 1730-1742.

[9] IEEE Standards Association, 03 de Agosto de 2015, http://standards.ieee.org/about/get/802/802.11.html;

[10] Chai Yan-Jie, Sun Ji-Yin, Gao Jing, Tao Ling-Jiao, Ji Jing, Bao Fei-Hu. 2008 International Conference on improvement of I2C Bus and RS-232 Serial Port under complex electromagnetic environment. Computer Science and Software Engineering, 2008. p. 178–81.

[11] Jongwoon Hwang, Woong Hee Kim, Sunyoung Lee, Real-time energy monitoring and controlling system based on Zigbee sensor Networks, Procedia computer science 5(2011) 794-797.

[12] Alena, R., Gilstrap, R., Baldwin, J., Stone, T., & Wilson, Fault tolerance in ZigBee wireless sensor networks. Aerospace Conference, 2011, 1–15

[13] Emily Gertz & Patrick Di Justo, Enviromental monitoring with arduino, O´Reilly 2011

[14] Sushabhan Choudhury, Piyush Kuchhal, Rajesh Singh, Anita,Zigbee and Bluetooth network based sensory data acquisition system, Procedia computer science 48(2015) 367-372.

[15] Anwari, M., Dom, M. M., & Rashid, M. I. M. Small scale PV monitoring system software design. Energy Procedia, 12(2011), 586–592.

[16] Telmo Moya Javier, Goglino Daniel Hoyos, Red de sensores y control inalámbrica para un sistema de generación de vapor solar térmico, Avances en energías renovables y medio ambiente, vol. 14, 2010. ISSN 0329-5184.

[17] Benghanem. Measurement of meteorological data based on Wireless data acquisition system monitoring. Applied Energy, 86(12), 2651–2660.

[18] M2M communications, 10 de Agosto de 2015, http://www.digi.com/en/

[19] M2M communications, 10 de Agosto de 2015, http://www.digi.com/support/k-base/k-baseresultdetl?id=2125

[20] Chen Thomas, Fu Zhi, He Liwen, Strayer Tim. Recent developments in network intrusion detection. IEEE Network 2009;23(1):4–5.

[21] Kuang-Yow Lian, Intelligent multi-sensor control system based on innovative technology integration via Zigbee and Wi-Fi Networks, Journal of network and computer applications, 36(2013) 756-767.

[22] Juan Jesús Almansa Madrigal, Desarrollo e implementación utilizando Arduino y Zigbee con un sensor ultrasónico para control de nivel de llenado, Departamento de ingeniería, electrónica, eléctrica y automática, Universidad de Rovira y Virgili, España. Septiembre de 2013.

[23] C#, 19 de Agosto de 2015, https://msdn.microsoft.com/es-es/library/kx37x362.aspx

[24] Latinoamérica renovable, 22 de septiembre de 2015, http://latinoamericarenovable.com/2012/09/24/mexico-y-sus-centros-de-investigacion-en-energias-renovables/

[25] ZigBee and Bluetooth Network based Sensory Data Acquisition System**,** Sushabhan Choudhury, Piyush Kuchhal,Rajesh Singh,Anita, University of Petroleum and Energy Studies, Dehradun, Bhubaneswar, Odisha, India, 2015.

[26] Red de sensores y control inalámbrico para un sistema de generación de vapor solar térmico, Telmo Moya Javier y Goglino Daniel Hoyos, Instituto Nacional de Energías No Convencionales, Argentina, 2010.

[27] Prototipo de red de sensores de temperatura, utilizando módulos Xbee, para control de incendios, Pedro Canché Martín, Mario Marín Montoya, Universidad del Caribe, Cancún, Quintana Roo, México, 2010.

[28] Las plantas de energía solar más grandes del mundo, 13 de Octubre de 2015, http://www.fierasdelaingenieria.com/las-plantas-de-energia-solar-mas-grandes-del-mundo/

[29] Solare datensysteme, 7 de Noviembre de 2015, http://www.solar-log.es/

[30] Solar energy monitoring, 11 de Noviembre de 2015, http://enphase.com/enlighten/

[31] Energía Alternativa de México, 9 de Diciembre de 2015, http://enalmex.com/

[32] Sistema de telemetría inalámbrica Mesh, 9 de Diciembre de 2015, http://www.comdiel.cl/sistema-de-telemetria-inalambrica-mesh-24ghz-wremote-s-1562.html

[33] Hsiao Sung-Jung, Fan Kuo-Chin, Sung Wen-Tsai, Ou Shih-Ching. Web-based system of pattern recognition for component patterns database by innovative algorithms. Malaysian Journal of Computer Science 2002;15(2):78–93.

CAPÍTULO 2. Fundamentos de comunicaciones inalámbricas

El propósito de un sistema de comunicación es el de transmitir información. Un sistema de comunicación comprende un transmisor, un canal sobre el cual la información se transmite, y un receptor para recoger la información. El canal de transmisión puede ser un simple par de conductores, un cable coaxial, una fibra óptica, una guía de ondas o el espacio libre [1].

El término radiofrecuencia, también conocido como espectro de radiofrecuencia se aplica a la porción menos energética del espectro electromagnético, situada entre los 3 Hz y 300 GHz. Las ondas electromagnéticas de esta región del espectro se pueden transmitir aplicando la corriente alterna originada en un generador a una antena. La primera red inalámbrica fue desarrollada en la Universidad de Hawaii en 1971 para enlazar los ordenadores de 4 islas sin utilizar cables de teléfono.

Las redes inalámbricas basadas en radio comenzaron a tener mucho éxito a principios de los 90 cuando la potencia de procesamiento de los chips llegó a ser suficiente para gestionar los datos transmitidos y recibidos a través de conexiones de radio [2].

2.1 Espectro Electromagnético.

El espectro Electromagnético es simplemente un nombre que los científicos han otorgado al conjunto de todos los tipos de radiación, cuando se los trata como grupo. La radiación es energía que viaja en ondas y se dispersa a lo largo de la distancia. La luz visible que proviene de una lámpara que se encuentra en una casa y las ondas de radio que provienen de una estación de radio son dos tipos de ondas electromagnéticas. Otros ejemplos son las microondas, la luz infrarroja, la luz ultravioleta, los rayos X y los rayos gamma [3].

Uno de los diagramas más importantes tanto en ciencia como en ingeniería es la gráfica del espectro electromagnético, dicho diagrama resume los alcances de las frecuencias o bandas que son importantes para comprender muchas cosas en la naturaleza y la tecnología, las ondas electromagnéticas pueden clasificarse de acuerdo a su frecuencia en Hz o a su longitud de onda. En la figura 2.1 se puede observar una imagen de cómo está dividido el espectro electromagnético y cuáles son los rangos en los que se encuentran las diferentes ondas [3].

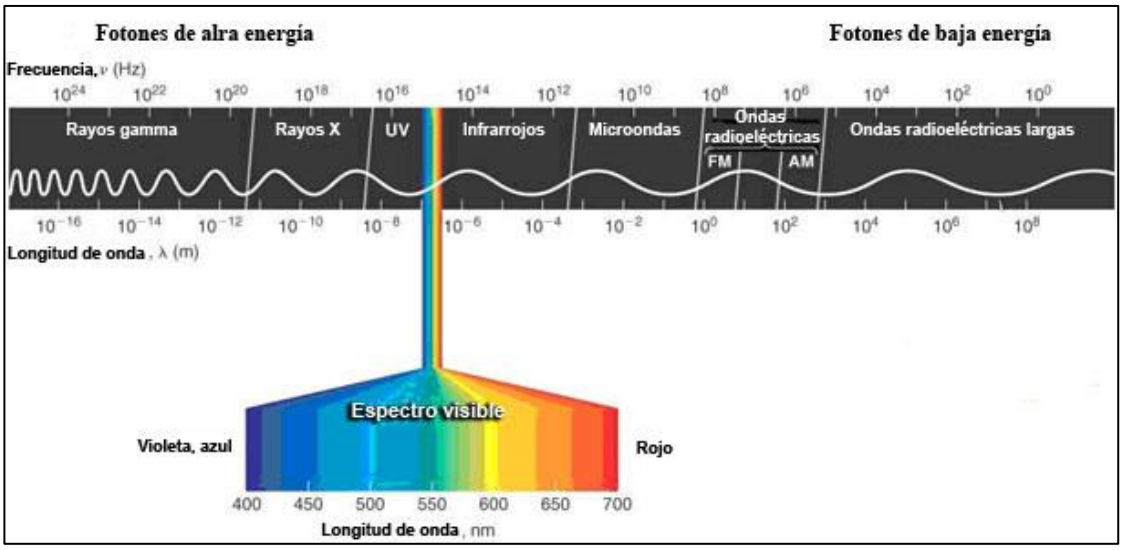

Figura 2.1. *Distribución del espectro electromagnético [4].*

Las ondas de radio son una porción del espectro electromagnético y sucede que la energía que transporta una onda electromagnética depende de la constante de Planck, la cual relaciona la energía de los fotones con la frecuencia de la onda lumínica según la siguiente fórmula [5]:

$$E = h \cdot c \tag{1}$$

Donde: E = Energía de los fotones

c = 299’792,458 m/s

h = 6.626 $J \cdot s$

Como se puede observar en la Tabla V, el espectro de radiofrecuencia es bastante amplio y abarca desde las frecuencias de los 3 Hz hasta los 300 GHz. Todos los rangos de frecuencia están destinados para un uso específico y los rangos comerciales son los que se encuentran entre los 30 kHz y los 30 GHz.

Tabla V. *Clasificación del espectro de radiofrecuencia [6].*

Nombre	Frecuencia	Longitud de onda	Uso
	< 3 Hz	> 100,000 Km	
Extra baja frecuencia	3 – 30 Hz	100,000 – 10,000 Km	No se utiliza en radiofrecuencia
Súper baja frecuencia	30 – 300 Hz	10,000 – 1,000 Km	Comunicaciones submarinas
Ultra baja frecuencia	300 – 3000 Hz	1,000 – 100 Km	Comunicaciones militares secretas
Muy baja frecuencia	3 – 30 kHz	100 – 10 Km	Comunicaciones militares y gubernamentales
Baja frecuencia	30 – 300 kHz	10 – 1 Km	Comunicaciones aéreas y marítimas
Frecuencia media	300 – 3000 kHz	1 Km – 100 m	Radiodifusión
Alta frecuencia	3 – 30 MHz	100 – 10 m	Seguridad, defensa, onda corta
Muy alta frecuencia	30 – 300 MHz	10 – 1 m	Televisión, radio FM, aviación, satélites, servicio marítimo
Ultra alta frecuencia	300 – 3000 MHz	1 m – 100 mm	Televisión, radiotransmisiones, uso personal, telefonía móvil, militar
Súper alta frecuencia	3 – 30 GHz	100 – 10 mm	Televisión vía satélite, radioenlaces, radar
Extra alta frecuencia	30 – 300 GHz	10 – 1 mm	Radioastronomía, radar de alta resolución
	> 300 GHz	< 1 mm	

Dentro del espectro electromagnético tenemos los rayos gamma de alta potencia que llegan de las estrellas, los rayos X, los microondas para cocinar nuestros alimentos, los infrarrojos que nos producen una sensación de calor, la luz visible que nos sirve para poder captar información y procesarla en el cerebro y tambien están las ondas de radio, las cuales tienen longitud de onda mas grande que las ondas de luz y muchas pueden viajar a través de materiales tales como la ropa, muebles o paredes.

Las ondas de radio se pueden propagar muy bien por el vacío y por lo tanto son ideales para las comunicaciones en las que la conexiones con cable no son tan prácticas, dichas ondas de radio ocupan un porción energética situada entre 300 kHz y 300 MHz.

2.1.1 Espectro electromagnético del Bluetooth, Wi-Fi y ZigBee

El rango de la zona en la que trabajan estas tecnologías abarca desde los 2,400 MHz hasta los 2,480 MHz, es decir, tienen un ancho de banda total de unos 80 MHz que pueden utilizar para modular su señal. Lo primero que hacen cada uno de ellos es repartirse esos 80 MHz en distintos canales en una técnica denominada *Frequency Division Multiple Access* (FDMA). Como se puede observar en la figura 2, cada tecnología lo hace de una forma [7].

Wi-Fi crea 12 canales de un ancho de banda de 22 MHz cada uno, por lo que necesariamente se sobreponen entre ellos, es decir, que los dispositivos que transmitan en bandas cercanas se van a producir interferencias uno a otro. Solo hay tres canales en los cuales no se producen, puesto que no existe solapamiento. Son los canales 1, 6 y 11, que son los canales que ocupan las redes.

El protocolo *ZigBee* crea 16 canales con un ancho de banda de 3 MHz por canal, por lo que pueden estar perfectamente distribuidos sin necesidad de interferirse entre ellos. Por otro lado, el *Bluetooth* usa 79 canales de 1 MHz cada uno [7], de

forma que tampoco hay interferencia entre sus canales y la señal que modulan en cada uno de ellos, esto se puede observar en la figura 2.2.

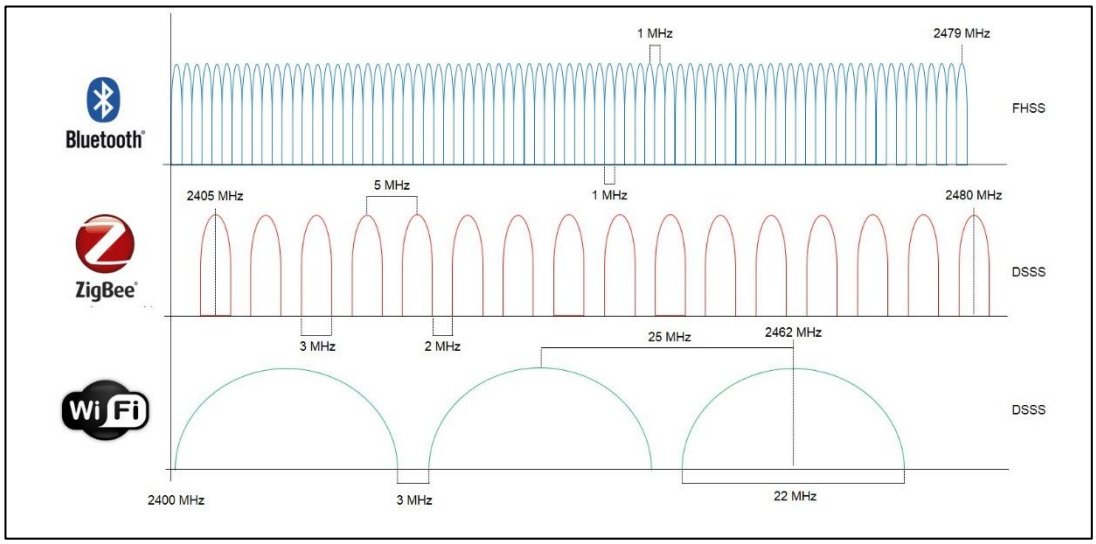

Figura 2.2. *Distribución del ancho de banda de diferentes tecnologías [7].*

2.1.2 Ecuación de transmisión en espacio libre

La problemática en las comunicaciones con radio es que requieren de mucha potencia en comparación con aquellas que viajan a través de cables, la razón de esto es fácil de entender, como la radio irradia desde una fuente hacia el espacio gran cantidad se atenúa sin que esta energía haya sido utilizada. Una buena comparación con esta idea podrían ser las ondas que se dispersan cuando tiramos una piedra en un estanque, lo mismo pasa con el sonido, ya que por ejemplo solo podemos entender un susurro desde una cierta distancia pero no es posible entenderlo si estamos a varios metros. Como ventajas los sistemas de radio requieren menos infraestructura, pueden desplegarse y configurarse más rápidamente que los sistemas basados en comunicaciones por cables [5].

Estos fenómenos se pueden explicar con la ecuación de transmisión en espacio libre [8], esta ley dice que para fenómenos ondulatorios tales como el sonido y la luz, la intensidad disminuye con el cuadrado de la distancia con respecto al punto en donde se origina, dicho de otra forma, cada vez que se dobla la distancia desde

la fuente, se requiere cuatro veces mas la cantidad de energía para mantener la señal.

Esto se puede observar en la siguiente figura:

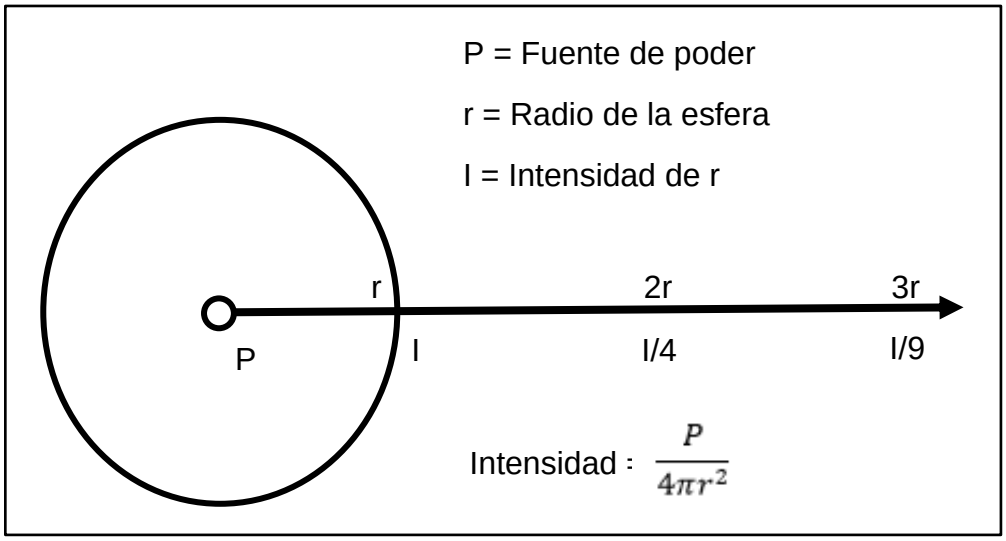

Figura 2.3. *Ley del cuadrado inverso [8].*

Esta ley puede ser explicada por la ecuación de transmisión de Friis:

$$P_{rx}(d) = \frac{P_{tx} * G_t * G_r * \lambda^2}{(4\pi d)^2} \quad (2)$$

Donde P_{tx} es la potencia de entrada, G_t y G_r son las ganancias de cada antena, λ es la longitud de onda y d es la distancia que hay entre las 2 antenas [8].

De hecho, las redes en malla *ZigBee* están diseñadas pensando en la ecuación de transmisión en espacio libre, los dispositivos *ZigBee* no necesitan baterías de alta capacidad para hacer llegar señales a grandes distancias, sino que en realidad hay muchos dispositivos en malla y cada uno de ellos hace de repetidor del siguiente para llegar al destinatario final con el objetivo de no desintegrar la señal de la fuente. De esta forma se pueden conseguir alcances grandes sin necesidad de utilizar elevadas potencias de transmisión que no serían viables por normativa y por consumo.

2.2 Antenas

Una antena es un dispositivo que sirve para transmitir y recibir ondas de radio. Convierte la onda guiada por la línea de transmisión (el cable o guía de onda) en ondas electromagnéticas que se pueden transmitir por el espacio libre, dependiendo de su forma y orientación, pueden captar diferentes frecuencias, así como niveles de intensidad.

Tipos

- **Omnidireccionales:** Es una antena constituida de un solo brazo rectilíneo irradiante en posición vertical y la señal que irradia es hacia los lados en forma de círculos pero no va hacia arriba o hacia abajo (ver figura 2.4), creando las ondas una especie de disco aplanado. Estas Antenas proporcionan un patrón de radiación horizontal de 360° y se utilizan éstas cuando la cobertura se requiere en todas las direcciones (horizontalmente) de la antena con los grados variables de cobertura vertical [9].

Figura 2.4. *Antena omnidireccional [9].*

- **Yagi:** Estas se componen de un arreglo de elementos independientes de antena, donde solo uno de ellos transmite las ondas de radio. El número de elementos (específicamente, el número de elementos directores) determina la ganancia y directividad, ver figura 2.5. Las antenas Yagi no son tan direccionales como las antenas parabólicas, pero son más directivas que las antenas panel [10].

Figura 2.5. *Antena Yagui [10].*

- **Parabólicas:** Las antenas parabólicas usan características físicas así como antenas de elementos múltiples para alcanzar muy alta ganancia y direccionalidad. Estas antenas usan un plato reflector con la forma de una parábola para enfocar las ondas de radio recibidas por la antena a un punto focal. La parábola también funciona para capturar la energía radiada por la antena y enfocarla en un haz estrecho al transmitir. Como puede verse en la Figura 2.6, la antena parabólica es muy direccional. Al concentrar toda la potencia que llega a la antena y enfocarla en una sola dirección, este tipo de antena es capaz de proveer muy alta ganancia [11].

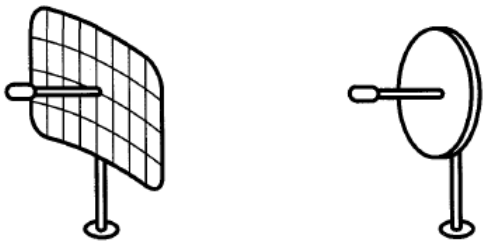

Figura 2.6. *Antenas parabólicas [11].*

2.3 Propagación de las ondas de radio

2.3.1 Propagación de radio frecuencia

El estudio de como las ondas electromagnéticas viajan e interactúan con la materia puede volverse extremadamente complejo, no obstante, existen varias simplificaciones importantes que pueden llevarse a cabo para estudiar más fácilmente las propiedades de las ondas electromagnéticas. En el vacío, las microondas de 2,4 GHz viajan a la velocidad de la luz. Una vez que se originan, estas microondas continuarán en la dirección en la cual fueron emitidas para

siempre, a menos que interactúen con alguna forma de materia. El rayo geométrico se utiliza para significar que las microondas están viajando en espacio libre. Puesto que las WLANs se encuentran usualmente en tierra, dentro de la atmósfera, las microondas viajan por el aire, no en el vacío.

De manera similar a la luz, cuando la radio frecuencia (RF) viaja a través de materia transparente, algunas de las ondas se ven alteradas. Por lo tanto, la velocidad de las microondas de 2,4 GHz y 5 GHz también cambia, a medida que las ondas viajan a través de la materia. No obstante, la cantidad de la alteración depende mucho de la frecuencia de las ondas y de la materia [12].

2.3.2 Refracción

Una superficie se considera lisa si el tamaño de las irregularidades es pequeño, en relación a la longitud de onda. De otro modo, se la considera irregular. Las ondas electromagnéticas se difractan alrededor de objetos interpuestos. Si el objeto es pequeño en relación a la longitud de onda, tiene muy poco efecto. La onda pasará alrededor del objeto sin perturbaciones. No obstante, si el objeto es grande, aparecerá una sombra detrás del mismo y una cantidad de energía significativa se refleja nuevamente hacia el origen. Si el objeto tiene alrededor del mismo tamaño que la longitud de onda, las cosas se complican, y aparecen patrones de difracción interesantes [13].

2.3.3 Reflexión

La reflexión tiene lugar cuando la luz rebota en la dirección general de la cual provino. Consideremos una superficie metálica lisa como interfaz. A medida que las ondas golpean la superficie, gran parte de su energía rebotará o se reflejará. Pensemos en experiencias comunes, como mirarse al espejo u observar la luz del sol reflejándose desde una superficie metálica o agua. Cuando las ondas viajan de un medio a otro, un determinado porcentaje de la luz se refleja. Esto se denomina reflexión de Fresnel.

Las ondas de radio también se reflejan al entrar en diferentes medios. La ley de reflexión puede describir estas reflexiones. Las ondas de radio pueden rebotar

desde diferentes capas de la atmósfera. Las propiedades reflexivas del área donde ha de instalarse la WLAN son extremadamente importantes y pueden determinar si una WLAN funciona o falla. Además, los conectores a ambos extremos de la línea de transmisión que se dirigen a la antena deberán estar apropiadamente diseñados e instalados, para que no tenga lugar ninguna reflexión de las ondas de radio. Si la línea y los conectores no coinciden apropiadamente, parte de la energía puede rebotar como eco y constituir una pérdida de potencia del sistema [13].

2.3.4 Difracción y dispersión

La dispersión de una onda en torno a un obstáculo se denomina difracción. Esta dispersión en ocasiones puede rodear un obstáculo. No obstante, para evitar una posible confusión con la refracción, que es un proceso enteramente diferente. Las ondas de radio pasan por una difracción a pequeña escala y a gran escala. Un ejemplo de difracción a pequeña escala son las ondas de radio de una WLAN que se dispersa en un ambiente interior. Un ejemplo de difracción a gran escala son las ondas de radio que se dispersan en torno a una montaña, hacia un área inaccesible. Como se puede observar en la figura 2.7 un efecto diferente tiene lugar cuando la luz golpea pequeñas partículas. Dependiendo de la frecuencia de la luz y del tamaño y la composición de las partículas, es posible un fenómeno denominado dispersión. La dispersión en general resulta en el redireccionamiento de la energía de onda entrante hacia direcciones que no son la dirección deseada [14].

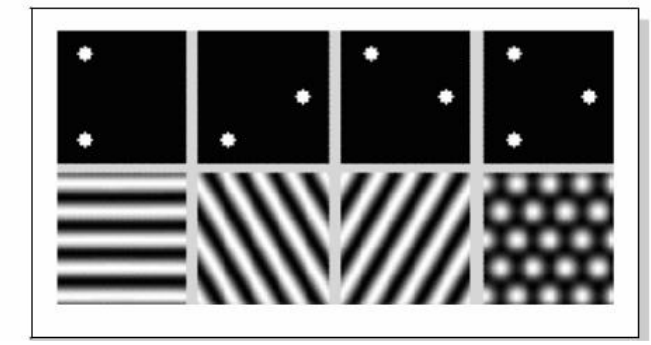

Figura 2.7. *Ejemplo de difracción [14].*

Un efecto diferente tiene lugar cuando la luz golpea pequeñas partículas. Dependiendo de la frecuencia de la luz y del tamaño y la composición de las partículas, es posible un fenómeno denominado dispersión. La dispersión en general resulta en el redireccionamiento de la energía de onda entrante hacia direcciones que no son la dirección deseada.

El sol irradia ondas visibles y otras ondas electromagnéticas, si no hubiera atmósfera, la luz llegaría directamente desde el sol y el resto del cielo estaría oscuro, excepto por las otras estrellas. Ésta es exactamente la visión que se obtiene desde la luna. Sin embargo, en la tierra el cielo es azul. Eso se debe a que las moléculas de la atmósfera dispersan la luz azul, mucho más que los otros colores. El resultado es que aunque la luz del sol de la mayoría de los colores llega directamente hacia un observador en la tierra, la luz azul se dispersa a través de una porción tan grande de la atmósfera que ésta aparece esencialmente azul brillante [14].

2.4 Tipos de señales

Una señal es una cantidad física que varía con el tiempo, el espacio o cualquier otra variable o variables independientes, la cual transporta información acerca de la naturaleza de un fenómeno físico. Matemáticamente una señal se describe como una función de una o más variables independientes, como se puede observar en la ecuación (3).

$$s(t) = 5t^2 \tag{3}$$

Donde:

s = la función que depende de (t)

t = la variable independiente

2.4.1 Señales en tiempo continuo y en tiempo discreto

Una señal *x(t)* se dice que será una señal en tiempo continuo si está definida para todo tiempo t. La figura 2.8 representa un ejemplo de una señal en tiempo continuo cuya amplitud o valor varía continuamente en el tiempo. Las señales en tiempo continuo surgen naturalmente cuando una forma de onda física tal como una onda acústica o una onda luminosa se convierte en una señal eléctrica [15].

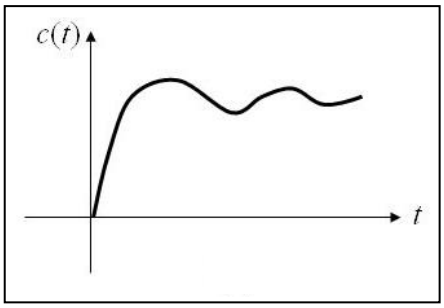

Figura 2.8. *Ejemplo de una señal en tiempo continuo [15].*

Por otra parte, una señal en tiempo discreto se define solo en instantes de tiempo discretos, dicha señal puede provenir de una señal en tiempo continuo que es muestreada a una tasa uniforme [16]. Una señal en tiempo continuo $x(t)$ en el tiempo $t=nJ$ produce una muestra de valor $X(nJ)$ como se puede observar en la siguiente ecuación.

$$X[n] = x(nJ), \quad n = 0, \pm 1, \pm 2, \ldots \tag{4}$$

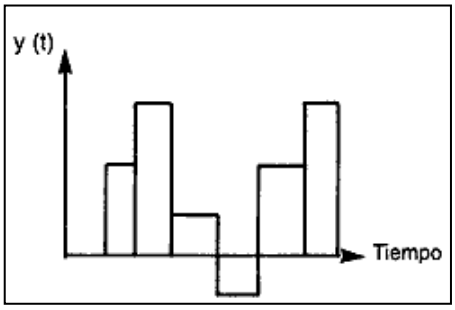

Figura 2.9. *Ejemplo de una señal en tiempo continuo [16].*

2.4.2 Señales pares e impares

Las señales pares son simétricas en torno al eje vertical u origen del tiempo tanto en una señal de tiempo continuo como en tiempo discreto y deben satisfacer la condición de la ecuación (5), en tanto que las señales impares son asimétricas en torno al origen del tiempo y deben satisfacer la ecuación (6).

$$x(-t) = x(t) \qquad para\ todo\ t \tag{5}$$

$$x(-t) = -x(t) \qquad para\ todo\ t \tag{6}$$

2.4.3 Señales periódicas y no periódicas

Una señal periódica $x(t)$ es una función que satisface la condición

$$x(t) = x(t+T) \qquad para\ todo\ t \tag{7}$$

Donde T es una constante positiva y esta define la duración de un ciclo completo de $x(t)$. y para toda señal que no hay valor de T que cumpla la condición de la ecuación (7) recibe el nombre de “no periódica” [16].

2.4.4 Señales determinísticas y aleatorias

Una señal determinística es aquella en torno a la cual no hay incertidumbre con respecto a su valor en cualquier instante de tiempo, en consecuencia, encontramos que dichas señales pueden modelarse como funciones de tiempo completamente especificadas. Por otra parte, una señal aleatoria es aquella en la que hay incertidumbre antes de su ocurrencia real, tal señal debe verse como parte de un todo o grupo de señales, con cada señal en el grupo con diferente forma de onda. El agrupamiento de tales señales se conoce como un proceso aleatorio. El ruido generado en el amplificador de un receptor de radio o televisión es un ejemplo de una señal aleatoria ya que su amplitud fluctúa entre valores positivos y negativos de un modo completamente aleatorio [16].

2.5 Sistemas de comunicación

Hay tres elementos básicos para todo sistema de comunicación, específicamente, *transmisor, canal y receptor*. El transmisor se localiza en un punto en el espacio, el receptor se ubica en algún otro punto separado del transmisor, y el canal es el medio físico que los conecta. El fin del transmisor es convertir la señal del mensaje producido por una fuente de información en una forma adecuada para la transmisión a través del canal, el mensaje de la señal podría ser una señal de voz, video o datos de computadora. Es posible que el canal sea una fibra óptica, un cable coaxial, un canal de satélite o un canal de radio móvil [17].

Un sistema de comunicación digital tiene un proceso complejo, el cual implica que el transmisor realice las siguientes operaciones para convertir la señal analógica en digital:

- **Muestreo**. Convierte la señal del mensaje en una secuencia de números, representando cada uno la amplitud de la señal del mensaje en un instante de tiempo particular.
- **Cuantización**. Implica representar cada número producido por el muestreador en el nivel más cercano elegido de un número finito de niveles de amplitud discretos. Por ejemplo, es posible representar cada muestra como un número binario de 16 bits, en cuyo caso hay 2^{16} niveles de amplitud.
- **Codificación**. Su propósito es representar cada muestra cuantizada mediante una palabra en clave integrada por un número finito de símbolos. Por ejemplo, en un código binario los símbolos pueden ser unos o ceros.

El transmisor tal vez implique operaciones adicionales, es decir, compresión de datos y codificación del canal, el objetivo de comprimir datos es quitar información redundante de la señal del mensaje y brindar de ese modo la utilización eficiente del canal mediante la reducción del número de bits/muestra requerido para la transmisión. La codificación del canal implica, por otra parte, la inserción de elementos redundantes en la palabra codificada de manera controlada; esto se realiza para proporcionar protección contra señales de ruido e interferencia

captadas durante el curso de la transmisión a través del canal. Por último, la señal codificada se modula sobre la onda portadora (usualmente senoidal) para la transmisión por el canal.

Las operaciones anteriores se llevan a cabo en el receptor en el orden inverso, de este modo se produce y entrega una estimación de la señal del mensaje original al destino del usuario. Hay dos modos básicos de comunicación, las cuales son la: ***transmisión***, la cual implica el uso de un potente transmisor y de numerosos receptores cuya construcción es relativamente económica, en este caso las señales que llevan información fluyen en una sola dirección. ***Comunicación punto a punto***, este proceso de comunicación tiene lugar a través de un enlace entre un solo transmisor y un solo receptor, en este caso suele haber un flujo bidireccional de señales que llevan información, hay un solo transmisor y un receptor en cada extremo del enlace [17].

2.6 Conclusiones

- El protocolo *ZigBee* crea 16 canales con un ancho de banda de 3 MHz por canal, por lo que pueden estar perfectamente distribuidos sin necesidad de interferirse entre ellos.
- Los dispositivos *ZigBee* no necesitan baterías de alta capacidad para hacer llegar señales a grandes distancias, sino que en realidad hay muchos dispositivos en malla y cada uno de ellos hace de repetidor del siguiente para llegar al destinatario final.
- La velocidad de transmisión de datos de los dispositivos *XBee* series 2 es de 250 kbps.
- Las antenas omnidireccionales son un tipo de antena que irradia su señal hacia todos lados en forma de círculos creando las ondas una especie de disco aplanado por lo que son el tipo de antena ideal para sistemas de monitoreo.

- Los elementos básicos para que todo sistema de comunicación pueda funcionar correctamente son específicamente el transmisor, canal y receptor, en este caso el transmisor y el receptor son los módulos *XBee* series 2 configurados como dispositivo final y coordinador, y el canal es el espacio aéreo en una frecuencia de 2.4 GHz.
- Un sistema de comunicación en malla es indispensable para realizar un flujo correcto de la información desde los dispositivos finales hasta el receptor.

Referencias

[1] Principios de las comunicaciones, José E. Briceño Márquez, tercera edición, 2005.

[2] Introducción a las redes inalámbricas, Adam Engst, Gleen Fleishman, ANAYA multimedia, 2010.

[3] J. Yang, C. Zhang, X. Li, Y. Huang, S. Fu, M.F. Acevedo. Integration of wireless sensor networks in environmental monitoring cyber infrastructure. *Wireless Networks, Springer*/*ACM*, Volume 16, Issue 4, pp. 1091-1108, May 2010.

[4] Scientific Committee on Emerging and Newly Identified Health Risks, Light Sensitivity (2008).

[5] Desarrollo e implementación utilizando Arduino y Zigbee con un sensor ultrasónico para control de nivel de llenado, Juan Jesús Almansa Madrigal, Universidad Rovira I Virgili Septiembre de 2013.

[6] Kuang-Yow Lian, Sung-Jung Hsiao and Wen-Tsai sung, "Mobile Monitoring and Embedded Control System for Factory Environment", www.mdpi.com/journal/sensors, December 2013.

[7] El Heraldo, 15 de Marzo de 2016, http://blogs.heraldo.es/ciencia/?p=1417

[8] Sistemas de comunicaciones electrónicas, Wayne Tomasi, Cuarta edición, Prentice Hall, 2003.

[9] J. Han, H. Lee, and K. Park. Remote-Controllable and Energy-Saving Room Architecture based on ZigBee Communication. IEEE Trans. On Consumer Electronics, Vol. 55, No. 1, pp. 264-268, Feb. (2009).

[10] K. Gill, S. Yang, F. Yao, and X. Lu. A ZigBee-based Home Automation System. IEEE Trans. On Consumer Electronics, Vol. 55, No. 2, pp. 422-430, May (2009).

[11] J. Lee, Y. Su, and C. Shen. A Comparative Study of Wireless Protocols: Blutooth, UWB, ZigBee, and Wi-Fi. The 33rd annual Conference of the IEEE Industrial Electronics Society(IECON), Nov. 5-8 (2007).

[12] Red de sensores Zigbbe, Ingeniería automática y electrónica industrial, Antón Girod Fortuño, Ramón Villarino Villarino.

[13] Emily Gertz, Patrick Di Justo, Enviromental Monitoring With Arduino, O'Reilly, 2011.

[14] K. Gill, S. Yang, F. Yao, and X. Lu. A ZigBee-based Home Automation System. IEEE Trans. On Consumer Electronics, Vol. 55, No. 2, pp. 422-430, May (2009).

[15] Tratamiento digital de señales, John G. Proakis, Dimitris G. Manolakis, 3 edición, Prentice Hall, Madrid 1998, ISBN 84-8322-000-8.

[16] Señales y sistemas, Haykin, Van Veen, Limusa Wiley, 2004.

[17] W.-S. Jang, W. M. Healy, and M. J. Skibniewski, "Wireless sensor networks as part of a web-based building environmental monitoring system," *Automation in Construction*, vol. 17, no. 6, pp. 729-736, Aug. 2008.

Capítulo 3. Metodología

El desarrollo del centro de monitoreo se llevó a cabo en un lapso de 18 meses, dicho proyecto se dividió en 3 etapas para poder realizarlo de manera ordenada. La primera etapa fue el estudio de los sistemas, la segunda fue la elaboración del software de visualización del centro de monitoreo y la tercera etapa fue la integración y el envío de los datos a internet.

Para el estudio de los sistemas se analizó el estado actual de 3 diferentes equipos y se instalaron los dispositivos necesarios para poder adquirir los datos de interés y al final de la etapa se realizaron las pruebas de correcto funcionamiento. En la segunda etapa se elaboró un programa de interfaz para adquirir los datos de manera inalámbrica provenientes de los módulos de radiofrecuencia y una base de datos donde se almacena toda la información adquirida.

En la etapa final se diseñó una página web y se integraron los sistemas para dar inicio al centro de monitoreo, con ello se tiene un monitoreo constante de los

sistemas y para poder visualizar la producción de energía de los sistemas no es necesario estar físicamente en el centro de monitoreo, sino que se puede realizar desde cualquier sitio que tenga acceso a internet ingresando a la página web del centro.

En la figura 3.1 se puede observar un esquema de la metodología utilizada en el proyecto y a continuación, se describen detalladamente cada una de las etapas y los equipos que se utilizaron para la realización satisfactoria del sistema de monitoreo inalámbrico.

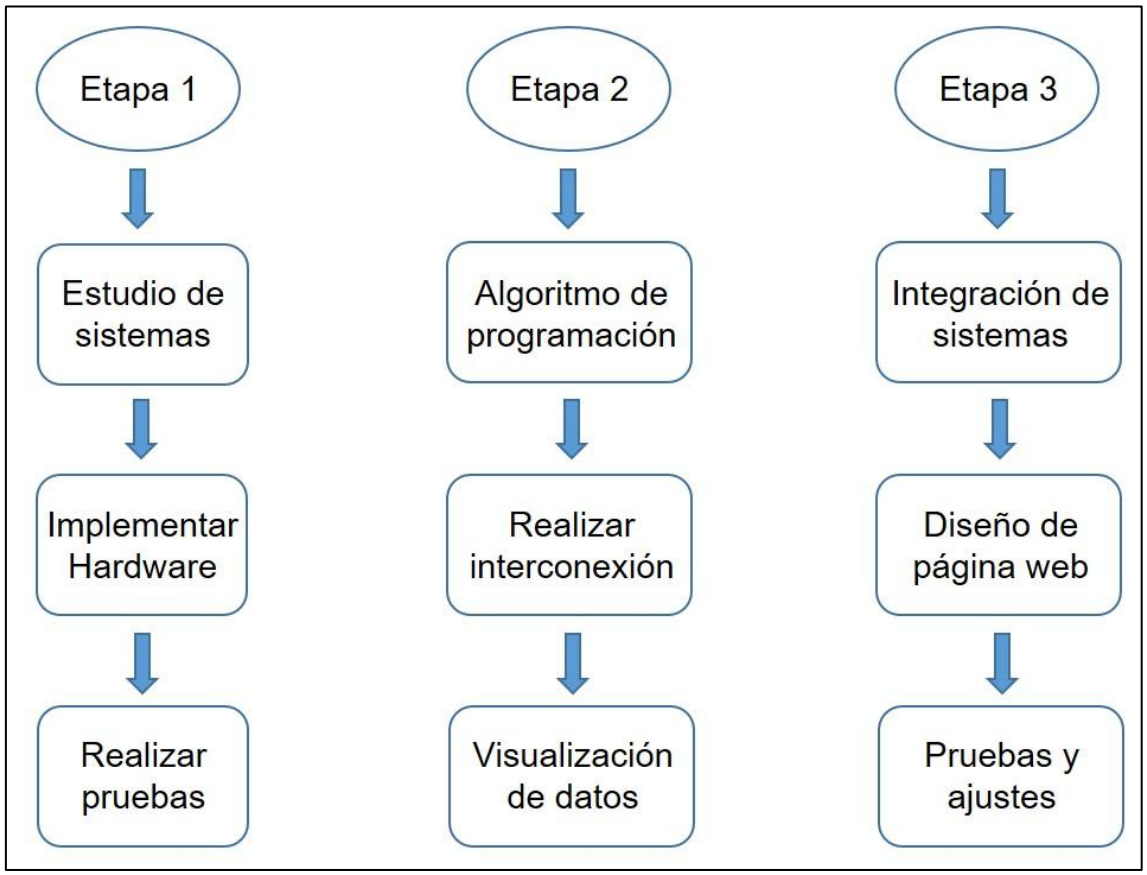

Figura 3.1. *Esquema representativo de la metodología utilizada en el proyecto.*

3.1 Sistemas energéticos renovables a monitorear

Los sistemas energéticos renovables seleccionados para monitorear fueron una central fotovoltaica autónoma de 2.5 kW, un deshidratador híbrido solar-eólico y un aerogenerador de 900 W, los cuales se encuentran instalados dentro de las

instalaciones del Centro de Investigación y Desarrollo Tecnológico en Energías Renovables (CIDTER) de la Universidad de Ciencias y Artes de Chiapas.

3.1.1 Central fotovoltaica de 2.5 kW

La central fotovoltaica de 2.5 kW instalada se utiliza para alimentar al Centro de Desarrollo y Evaluación de Biodigestores dentro del CIDTER, dicha central está constituida por 10 paneles de 31.73 V_{DC} y 7.88 A con capacidad de 250 W cada panel, los cuales están conectados en un arreglo serie paralelo como se puede observar en la figura 3.2 para generar un voltaje máximo aproximado de 64 V_{DC} y una corriente máxima de 40 A [1].

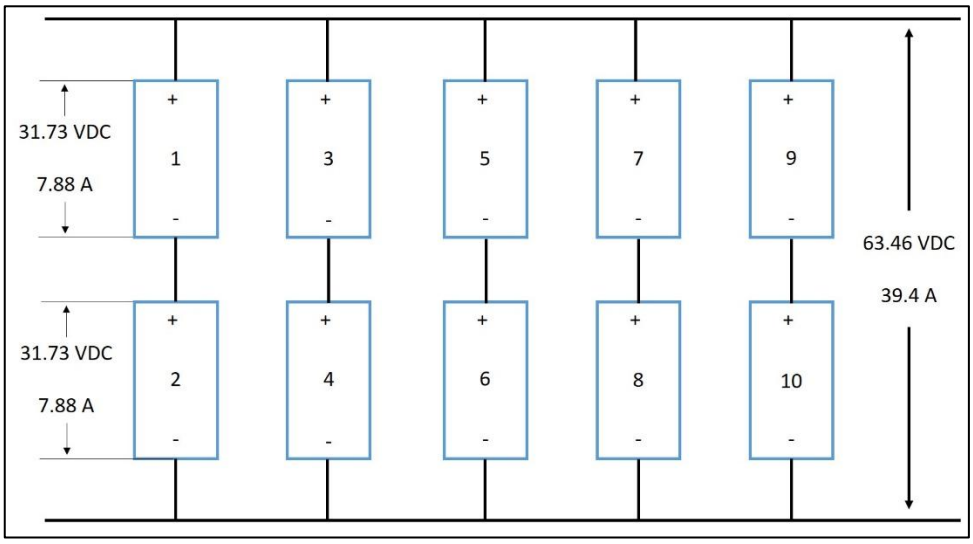

Figura 3.2. *Arreglo serie-paralelo de la central fotovoltaica de 2.5 kW.*

El sistema instalado consta de 10 paneles solares de 250 W, un controlador de carga de 150 V_{DC} máximo y 64 A_{DC} máximo, un inversor con salida monofásica de 120 V_{AC} a 60 Hz y potencia de 2500 W y un banco de baterías de 4 piezas de 6 V_{DC} y 390 A/Hora conectadas en serie para poder tener un voltaje de carga de 24 V_{DC}.

De acuerdo al estudio realizado en dicha central fotovoltaica se concluyó que los equipos necesarios para poder monitorear de manera inalámbrica al sistema son: 1 divisor de voltaje, 1 sensor de voltaje de 25 V_{DC} máximo, 1 sensor de corriente de 50 A máximo, un medidor de energía consumida (kilowattimetro), una placa Arduino y un módulo de radiofrecuencia *XBee* series 2.

3.1.2 Aerogenerador de 900 W

El aerogenerador de 900 W instalado se utiliza para alimentar a un deshidratador híbrido solar-eólico, dicho equipo puede producir energía a 12, 24, 36 y 48 V_{AC}. Los componentes con los que cuenta dicho sistema son un controlador de carga configurable en el voltaje de entrada de 12 hasta 48 V_{AC}, un sistema de resistencia para frenar el aerogenerador en caso de que se excedan las revoluciones por minuto que permite dicho equipo cuando se tengan velocidades de viento muy altas, un banco de baterías de 12 V_{DC} de 3 piezas conectadas en paralelo y un inversor con voltaje de entrada de 12 V_{DC} y voltaje de salida de 120 V_{AC} monofásico, con 1500 W de potencia.

3.1.3 Deshidratador híbrido solar-eólico

El deshidratador híbrido solar-eólico está constituido por un sistema complejo de componentes los cuales son: tanque de almacenamiento de agua que provee a un sistema de tubos al vacío para calentarla, una bomba que recircula el agua caliente por la cámara de secado, un radiador por el cual pasa el agua caliente, un ventilador que recircula el aire caliente dentro de la cámara, un extractor para eliminar la humedad y un sistema de calentamiento auxiliar que consta de resistencias eléctricas para ocasiones en las cuales la radiación no sea la suficiente para calentar el agua y alcanzar la temperatura deseada dentro de la cámara de secado o para cuando se necesite deshidratar producto por las noches. El sistema de control consta de un sensor de temperatura del agua del tanque, un sensor de temperatura y humedad de la cámara de secado, una placa Arduino, indicadores luminosos para visualizar el estado de los equipos (si están encendidos o apagados) y una etapa de potencia.

3.2 Selección e implementación de dispositivos

De acuerdo a los resultados obtenidos del análisis y el estudio del estado actual de los sistemas seleccionados para monitorear se eligieron los sensores y dispositivos adecuados para cada sistema. Dichos componentes se eligieron de acuerdo a sus características con respecto a otros, tomando en cuenta su funcionamiento, accesibilidad en el mercado, tamaño, configuración y costo.

Los dispositivos y sensores utilizados en cada sistema se describen a continuación:

- **Central fotovoltaica de 2.5 kW:** Se diseñó un circuito divisor de voltaje, un circuito para alimentación y comunicación por puerto serial del dispositivo de radiofrecuencia, se instalaron: un sensor de voltaje de 25 V_{DC} máximo, un sensor de corriente de 50 A, una placa Arduino Mega, un medidor de energía consumida (kilowattimetro) y módulo de radiofrecuencia *XBee* series 2.
- **Deshidratador híbrido solar-eólico:** Debido al grado de automatización con el que cuenta dicho sistema solamente fue necesario instalar un módulo de radiofrecuencia *XBee* series 2 y la modificación del código en una parte del programa de control del deshidratador. El sistema ya cuenta con los sensores y dispositivos necesarios para monitorear el equipo (sensor de temperatura y humedad de la cámara de secado, sensor de temperatura del tanque de almacenamiento de agua y una placa Arduino).
- **Aerogenerador de 900 W:** Los equipos que se necesitaron para el aerogenerador son muy similares a los utilizados en la central fotovoltaica de 2.5 kW, se diseñó un circuito para alimentación y comunicación por puerto serial del dispositivo de radiofrecuencia, se instalaron: un sensor de voltaje de 25 V_{DC} máximo, un sensor de corriente de 50 A, una placa Arduino Mega, un medidor de energía consumida (kilowattimetro) y módulo de radiofrecuencia *XBee* series 2.

3.2.1 Sensor de voltaje

El sensor de voltaje [2] utilizado para la central fotovoltaica de 2.5 kW y el aerogenerador de 900 W fue un sensor que es capaz de leer desde 0.02445 hasta 25 V_{DC} y mide el voltaje basándose en el principio de diseño de divisor de tensión resistivo, con una resolución de 0.00489 V_{DC} y un error del 2.4%. Dicho sensor consta de tres pines de los cuales dos son de alimentación (“+” = 5 V_{DC} y “-“ = GND) y uno es la señal de salida (S), Además de 2 bornes en los cuales se conecta el positivo (+) y el negativo (-) del voltaje a leer. Dicho sensor se puede observar en la figura 3.3.

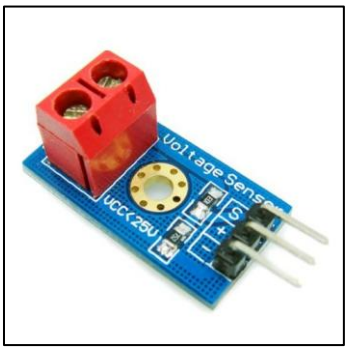

Figura 3.3. *Sensor de voltaje de 25 V_{DC} máximo [2].*

3.2.2 Sensor de corriente

El sensor utilizado en ambos sistemas fue un sensor de corriente [3] de 50 A (como se puede ver en la figura 3.4) que es capaz de leer corriente alterna y corriente directa utilizando el efecto Hall, dicho equipo se alimenta con 5 V_{DC} y su sensibilidad es de 400 mV/A, puede trabajar en un rango de temperatura de -40 ºC a 150 °C y tiene un error del 1.2%.

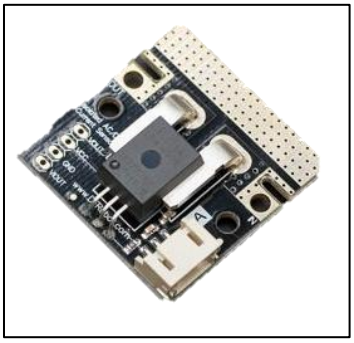

Figura 3.4. *Sensor de corriente de 50 A en AC Y DC máximo [3].*

3.2.3 Arduino Mega 2560

Para el control de los sensores se utilizó una placa de Arduino Mega 2560 [4], la cual cuenta con un microcontrolador ATmega1280, trabaja con un voltaje de 5 V_{DC}, cuenta con 54 pines de entradas/ salidas (I/O) digitales, de las cuales 15 son con salida PWM para control, 16 pines de entradas analógicas, la corriente máxima que puede tener en sus entradas analógicas es de 40 mA, cuenta con una memoria flash de 128 Kb y una velocidad de procesamiento de 16 MHz, en la figura 3.5 se puede observar una placa Arduino Mega 2560.

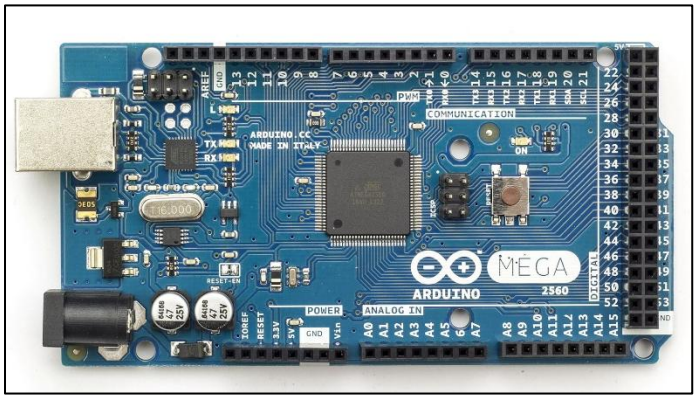

Figura 3.5. *Placa Arduino Mega 2560 utilizada en el proyecto [4].*

3.2.4 Medidor de energía consumida

El dispositivo utilizado fue un kilowattimetro [5] (figura 3.6) que puede medir el consumo de energía en corriente alterna, dicho modelo puede trabajar a 120 V_{AC} o 240 V_{AC} monofásico y 65 A máximo, el medidor muestra en tiempo real parámetros como el voltaje, corriente, frecuencia y factor de potencia. Además de que genera 1600 impulsos por kilowatt consumido, los cuales se pueden enviar al Arduino para ir contabilizando la energía consumida. Para realizar estas mediciones se tiene que conectar en serie al sistema que va a monitorear, es decir, la línea del sistema se conecta a una terminal del medidor y la salida de medidor se regresa al sistema, además de que se conecta el neutro, para poder leer los impulsos se conecta una señal a la terminal 3 del equipo y la terminal 4 que es la salida se envía de regreso al Arduino.

Figura 3.6. *Medidor de energía consumida 120 V_{AC} – 240 V_{AC} [5].*

3.2.5 Módulo de radiofrecuencia

Para la etapa de comunicación inalámbrica se utilizó un módulo de radiofrecuencia *XBee* series 2 [6], el cual envía los datos por el puerto serial con un alcance máximo de 120 m a campo abierto, la frecuencia a la cual trabajan es de 2.4 GHz y su velocidad de transmisión es de 250 kbps, se pueden alimentar a 3.3 o 5 V_{DC} y el consumo potencia de éstos módulos es de 2 mW. Consta de 20 pines, 2 de alimentación, 2 de comunicación, 1 de reset, 1 reservado, 6 de 10 bits para entrada ADC y 8 pines de entradas/salidas digitales.

Figura 3.7. *Módulo de radiofrecuencia XBee series 2 [6].*

3.2.6 Calibración de sensores

Para poder realizar un adecuado sistema de monitoreo de cualquier dispositivo es necesario calibrar los sensores mediante los cuales se obtendrán los valores de los parámetros deseados, por lo cual después de haber elegido cada uno de los sensores a utilizar en los diferentes sistemas a monitorear, se realizó la etapa de calibración de dichos sensores.

- **Central fotovoltaica de 2.5 kW y Aerogenerador de 900 W**
  Los sensores de voltaje y corriente utilizados en la central fotovoltaica fueron calibrados utilizando como patrón de referencia un multímetro digital de banco de la marca *KEITHLEY* modelo 2000 con un error de 0.02% (certificado de calibración PC2KXGRK78) y comparados con los valores visualizados por el controlador de carga de la marca *OutBack Power systems* modelo FW250 instalado en el sistema. Los sensores utilizados en el

aerogenerador fueron calibrados utilizando únicamente como referencia el multímetro de banco mencionado anteriormente.

- **Deshidratador híbrido solar-eólico**

  Los sensores utilizados para el deshidratador fueron un sensor de temperatura y humedad y un termopar, los cuales fueron calibrados utilizando como patrón de referencia un sensor de temperatura y humedad de la marca *Measurement Computing* modelo USB-502-LCD el cual tiene un error del 1.1% con un rango de medición de -35 a +80 °C y del 0 a 100 % de humedad.

3.2.7 Arduino Software

Los dispositivos Arduino tienen su propio software de programación para poder configurarlos y cargarle programas dependiendo de la aplicación que se tenga. El entorno de programación es secuencial y utiliza el lenguaje “C” y consta de dos partes importantes, la primera es la de configuración de los parámetros a utilizar (se ejecuta una sola vez en el programa) donde se definen los pines de entrada y salida del Arduino, la velocidad de transmisión mediante el puerto serial, así como otros más. Esto se realiza dentro de la sección llamada “***void setup ()***” y la segunda es el ciclo de trabajo que realizará constantemente el Arduino [7] donde se le indica que esté leyendo a cada determinado tiempo las entradas destinadas para los sensores, así como enviando la información por el puerto serial hacia el módulo de radiofrecuencia, esto se realiza dentro de la sección “***void loop ()***”. La programación del Arduino que monitorea al aerogenerador se puede observar en el Anexo “A”, así como los códigos de los arduinos de la central fotovoltaica y deshidratador.

3.2.8 XCTU Software

Para poder configurar los módulos de radiofrecuencia fue necesario utilizar el software de programación de la empresa *DIG*I llamado *XCTU* [8], mediante el cual se pueden programar y reprogramar los chips de dichos módulos dependiendo del tipo de comunicación que se requiera, ya sea punto a punto, punto a multipunto, que módulo será el coordinador y que módulos serán los *routers* o dispositivos finales, si van a utilizar el modo de transmisión transparente (AT) o transmisión de paquetes (API), entre muchos otros parámetros. Se programó un módulo como

coordinador y 3 módulos como dispositivos finales, todos en modo AT y con el mismo PAN ID o canal de trabajo para que se puedan comunicar entre sí [8].

3.3 Programa de interfaz del centro de monitoreo

Una vez instalados los sensores y dispositivos necesarios para adquirir los datos de cada sistema monitoreado, se necesitaba elaborar un programa de interfaz que fuera capaz de recibir los datos enviados por los módulos de radiofrecuencia, que muestre los valores y los grafique en tiempo real y además que esté guardando toda esa información en una base de datos exclusiva del centro de monitoreo para visualizar el comportamiento de los sistemas en cualquier día. Por lo que fue necesario elaborar una lógica del programa que realizaría todas estas tareas.

Después de ello se realizó la escritura del código de programación en el programa *Microsoft Visual C# 2010 Express* ya que ofrece un entorno de trabajo gráfico, dicho programa recibe los datos provenientes de los diferentes sistemas, los visualiza, los grafica, los separa y además de ello envía a la base de datos dicha información para almacenarla; para realizar esta conexión es necesario tener instalado y activado el programa *WampServer64*, el cual se encarga de establecer un canal por el cual se comunica el software de visualización con la base de datos.

Dicha base de datos se realizó en el programa *MySQL Workbench 5.2 CE*, el cual también tiene un entorno gráfico y no requiere demasiados comandos ni código para realizar la base de datos, únicamente se crea una base de conexión con un nombre determinado y las respectivas tablas donde se almacenará la información. El usuario tiene la posibilidad de visualizar en cualquier momento cada una de las tablas y guardar esa información en archivos separados por comas (.CSV).

3.3.1 Algoritmo de programación

Antes de comenzar con la programación en *Microsoft Visual C# 2010 Express* se elaboró un algoritmo de programación con el cual el programa del centro de monitoreo fuera capaz de realizar todas las tareas asignadas. La lógica se puede observar en la figura 3.8 y fue la siguiente: se inicia el programa, busca los puertos COM (puerto serial de la computadora) disponibles, si no encuentra ningún puerto

COM disponible, envía un mensaje de error y cierra el programa, si detecta uno o más puertos disponibles los visualiza y permite elegir el puerto con el cual se va a trabajar; si el puerto seleccionado no es correcto o no funciona, regresa a la etapa de buscar puertos disponibles; si el puerto es correcto, muestra un mensaje de "conexión lista", y luego inicia el monitoreo, revisa la conexión a la base de datos, si no hay conexión a la base de datos causa un error, cierra el programa e indica que hay que iniciar el programa que establece la conexión entre *Microsoft Visual C# 2010 Express* y la base de datos, dicho programa es *WampServer64*; si la conexión se establece con éxito, envía un mensaje de "conectado" y realiza la conexión al servidor web para poder enviar la información a internet, si no hay conexión envía un error y se debe iniciar el servidor para poder tener acceso al centro de monitoreo desde internet; luego comienza a leer el puerto serial para ver si recibe información, si no recibe datos continúa leyendo hasta que lee uno o más datos, cuando lee los datos, los separa de acuerdo a su codificación, los visualiza, los grafica, envía dicha información a la base de datos y al mismo tiempo envía la información a la página web del centro de monitoreo, este ciclo lo realiza continuamente sin interrupción, si el usuario decide cerrar el programa, *Microsoft Visual C# 2010 Express* cierra todas las conexiones y con ello termina el proceso de monitoreo.

3.3.2 Microsoft Visual C# 2010 Express

Este programa está desarrollado en un entorno integrado para sistemas operativos Windows orientado a objetos diseñado para la infraestructura de lenguaje común, Su sintaxis básica deriva de *C/C*++ y utiliza un modelo de objetos de la plataforma *.NET*, similar al de *Java*.

Dicho programa tiene la facilidad de insertar objetos, cuadros, botones, comboboxes, entre muchos otros y al hacer doble click sobre cada uno de ellos se puede acceder a la parte del programa para indicarle que deberá hacer cada objeto, además de que se puede establecer una conexión con el puerto serial de la computadora de manera muy sencilla, permitiendo tener un programa no tan complejo como lo sería en otra plataforma de programación como lo es Java [9].

En el anexo “B” se puede observar el código del programa de adquisición de datos del centro de monitoreo.

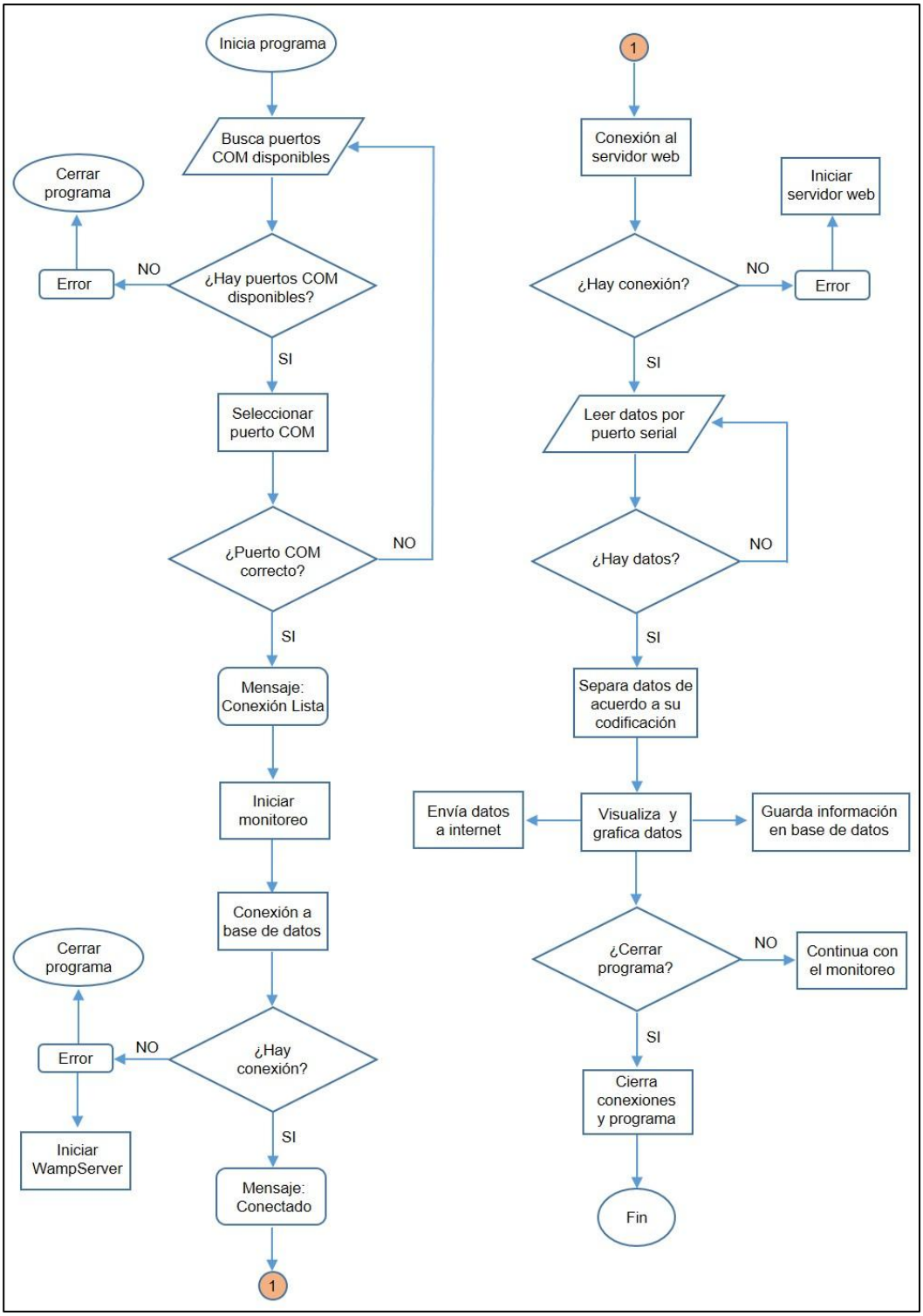

Figura 3.8. *Diagrama de flujo del algoritmo de programación utilizado.*

3.3.3 WampServer64

WampServer64 es un entorno para el desarrollo de páginas web en Windows que puede desarrollar aplicaciones en *PHP*, *APACHE*, MySQL y que además incluye herramientas para manejar bases de datos de manera rápida y sencilla. En la aplicación del proyecto se utiliza para establecer la conexión entre *Microsoft Visual C# 2010 Express* y *MySQL Workbench 5.2 CE* que es el programa donde se encuentra la base de datos. Lo único que se necesita hacer es iniciar *WampServer64* y revisar que el servicio de *MySQL* se encuentre activo, si no lo está hay que activarlo para que nos permita conectar ambos programas. En la figura 3.9 se puede observar el entorno de dicho programa [10].

Figura 3.9. *Entorno del programa WampServer64 [10].*

3.3.4 MySQL Workbench 5.2 CE

El programa elegido para realizar la base de datos fue *MySQL Workbench 5.2 CE* desarrollado por *ORACLE*, el cual ofrece múltiples herramientas para el diseño sencillo de bases de datos pudiendo realizarlas desde en entorno de programación o desde un entorno gráfico para crear tablas con sus respectivos campos [11]. Se creó una conexión con el nombre de “Centro-de-Monitoreo” con un determinado usuario y un *Host*, dentro de ella se elaboró la base de datos llamada historial, la cual contiene 3 diferentes tablas llamadas aerogenerador, central y secador, una para cada sistema monitoreado.

3.4 Integración de sistemas del centro de monitoreo

La tercera etapa con la cual concluye el proyecto es la integración de los sistemas monitoreados con el servidor web para poder enviar la información en tiempo real a internet, en la página web del centro de monitoreo se puede visualizar en tiempo real los parámetros de cada sistema, así como sus respectivas gráficas. El programa utilizado es *Adobe Dreamweaver*, el cual realiza una conexión entre la base de datos en tiempo real y la página web para poder visualizar la información en internet. Con ello no es necesario encontrarse dentro del centro de monitoreo para poder ver el comportamiento de los sistemas, sino que se puede tener acceso desde cualquier equipo de cómputo en cualquier parte del mundo que tenga acceso a internet, únicamente ingresando a la página web del centro de monitoreo.

3.4.1 Adobe Dreamweaver

Este programa es una aplicación destinada a la construcción, diseño y edición de sitios, videos y aplicaciones web basados en estándares. La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización, puesto que en este programa sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en *Javascript-C*, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de *C++* sino rutinas de *Javascript* que hace que sea un programa muy fluido y todo ello permite que programadores y editores web hagan extensiones para su programa y lo pongan a su gusto [12]. En el anexo “C” se puede observar el código en *PHP* de la página web del centro de monitoreo.

Figura 3.10. *Logo del software Adobe Dreamweaver [12].*

3.5 Conclusiones

- Se realizó un estudio de los sistemas energéticos seleccionados determinando las condiciones en las que se encontraba cada uno de ellos.
- Se adquirieron e instalaron los equipos indispensables para la obtención de los parámetros necesarios de cada sistema.
- Se calibraron los sensores usados en cada sistema para tener mediciones fiables.
- Se elaboró un algoritmo de programación para el programa de adquisición y visualización de datos del centro de monitoreo.
- La información adquirida por el módulo de radiofrecuencia se almacena en una base de datos.
- Se diseñó una página web para la visualización de los sistemas en tiempo real en internet.
- Se integraron los sistemas monitoreados almacenando la información adquirida en el centro de monitoreo.

Referencias

[1] Ana Cristina Peláez Hernández, Estudio de rendimiento de una central fotovoltaica conectada a la red, Universidad de Ciencias y Artes de Chiapas. Octubre de 2015.

[2] Arduino voltage sensor, 27 de Mayo de 2016, http://www.instructables.com/id/Arduino-Voltage-Sensor-0-25V/

[3] DFRobot, 1 de Junio de 2016, http://www.dfrobot.com/image/data/SEN0098/ACS758%20datasheet.pdf

[4] Lien Chia-Hung, Bai Ying-Wen, Lin Ming-Bo. Remote controllable power outlet system for home power management. IEEE Transactions on Consumer Electronics 53(2007). 1634–1641.

[5] Medidor eléctrico monofásico, 13 de Junio de 2016, http://chinaenergymeter.es/1-1-1-sw-single-phase-electricity-meter.html

[6] Yuksekkaya B, Kayalar AA, Tosun MB, Ozcan MK, Alkar AZ. A GSM, internet and speech controlled wireless interactive home automation system. IEEE Trans- actions on Consumer Electronics 2006;52(3):837–843.

[7] Yang Yu-Chen, Wang Sheng-Tien, Tseng Yi-Jen, Koong Lin Hao-Chiang. Light up! Creating an interactive digital artwork based on Arduino and Max/MSP design. In: 2010 International computer symposium (ICS); 2010. p. 258–263.

[8] Xin Wang, Longquan Ma, Huizhong Yang, Online water monitoring system based on zigbee and GPRS, Procedia Engineering 15(2011). 2680-2684.

[9] Software Engineering with Microsoft Visual Studio Team System, Juan J. Pérez, Sam Guckenheimer, Addison-Wesley Professional, 2006.

[10] Khushali Pandit, Varsha Bhosale, Implementation of Location based Steganography on mobile Smartphone using Android Platform, International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 2606-2609

[11] Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, “Digital image steganography: Survey and analysis of current methods,” Signal Processing, vol. 90, no. 3, pp. 727–752, 2010.

[12] Farihah Shariff, Nasrudin Abd Rahim, Hew Wooi Ping, Zigbee-based data acquisition system for online monitoring of grid-connected photovoltaic system, Expert systems with applications 42(2015) 1730-1742.

Capítulo 4. Resultados

El objetivo principal de este proyecto ha sido el desarrollo de un centro de monitoreo para poder llevar un histórico del comportamiento de los diferentes sistemas energéticos renovables seleccionados y con ello realizar las acciones que sean necesarias para el buen funcionamiento y la máxima eficiencia de cada sistema, para lo cual se tuvieron que cumplir 3 objetivos particulares. Este trabajo es el inicio de un sistema de monitoreo más complejo que en un futuro cercano también permita controlar de manera remota los sistemas energéticos renovables.

La primera etapa del proyecto consistió en realizar un estudio del estado actual de los sistemas seleccionados a monitorear, los cuales fueron un deshidratador híbrido solar – eólico, una central fotovoltaica de 2.5 kW y un aerogenerador de 900 W.

La segunda etapa fue el diseño del software de visualización del centro de monitoreo mediante el cual se pueden visualizar los datos obtenidos en tiempo real de cada sistema, así como sus respectivas gráficas, para el cual se utilizó el software *Microsoft Visual C# 2010 Express*. Además de la elaboración de una base de datos realizada en *MySQL Workbench 5.2 CE*.

La etapa final del proyecto fue la integración de los sistemas para el envío de la información a internet, es decir, se diseñó una página web en *Adobe Dreamweaver* para interconectarla a la base de datos y así enviar la información al servidor de la Universidad de Ciencias y Artes de Chiapas.

4.1 Deshidratador híbrido solar - eólico

- Se modificó la programación de la placa Arduino que controla al deshidratador para poder realizar el envío de datos por el puerto serial con una codificación específica del sistema. La codificación consiste en letras iniciales al envío de datos por puerto serial como se puede ver a continuación.

```
Serial.print("ST.");
Serial.print(temperature, 1);
Serial.println(".C");
Serial.print("SH.");
Serial.print(humidity, 1);
Serial.println(" %");
Serial.print("STT.");
Serial.print(temptanq, 1);
Serial.println(".C");
```

El valor de la temperatura de la cámara de secado del deshidratador se envía con las letras iniciales “ST” seguido del valor obtenido del sensor (variable con el nombre “*temperature*”) y al final de la línea se le agrega la unidad de medida (°C), la humedad se envía con las letras “SH” seguido del valor obtenido por el sensor (variable con el nombre “*humidity*”) con su respectiva unidad de medida al final (“%) y la temperatura del tanque de almacenamiento con las letras “STT” seguido del valor del sensor (variable con el nombre “*temptanq*”) y al final su unidad (°C).

- Se diseñó un circuito electrónico para la alimentación y el envío de datos del dispositivo de radiofrecuencia, como se puede observar en la figura 4.1 el módulo de radiofrecuencia funciona con un voltaje de alimentación de 3.3 V_{DC} a 5 V_{DC} y gracias al diseño del circuito dicho módulo se puede alimentar

desde 3.3 V_{DC} hasta 24 V_{DC} ya que tiene un regulador de voltaje de 24 V_{DC} a 5 V_{DC}. El módulo de radiofrecuencia se conecta al Arduino mediante los pines TX y RX para llevar a cabo la comunicación.

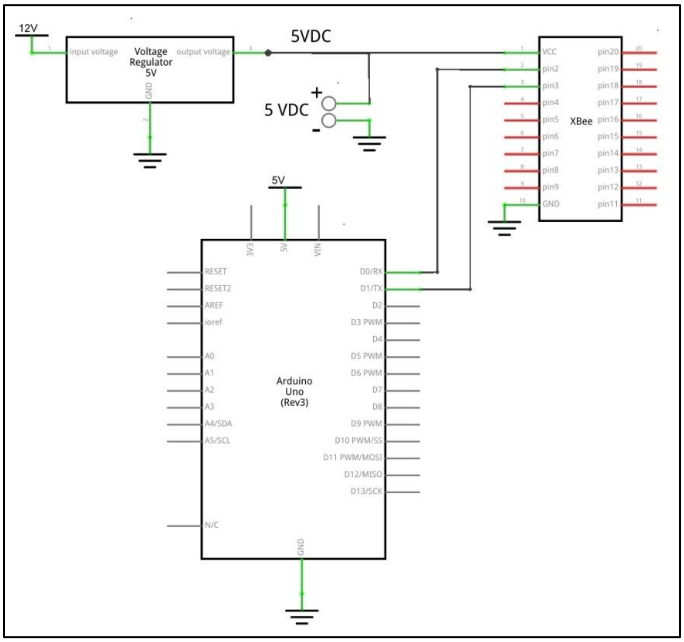

Figura 4.1. *Diseño del circuito de alimentación para el dispositivo de radiofrecuencia.*

- Se instaló un módulo *XBee* series 2 para la comunicación inalámbrica por medio de radiofrecuencia.
- Se obtuvo un rango de cobertura de 50 m a campo abierto entre el deshidratador y el centro de monitoreo.

4.2 Central fotovoltaica de 2.5 kW

- Se diseñó un circuito electrónico para la lectura del voltaje y corriente producidos por los paneles solares, así como para el consumo de potencia en corriente alterna, además de la alimentación de los sensores utilizados, como se puede observar en la figura 4.2.

El circuito de monitoreo se conecta en paralelo con el banco de baterías en el controlador de carga, el sensor de corriente se conecta en serie con la entrada de los paneles, la salida de dicho sensor se envía a un divisor de voltaje y la salida de dicho divisor se conecta el sensor de voltaje. Las señales de salida del sensor de

corriente y voltaje van a las entradas analógicas A0 y A1 de la placa Arduino respectivamente.

Los pines 2 y 3 del módulo *XBee* se conectan al pin RX y TX de la placa Arduino respectivamente para el envío de los datos de manera inalámbrica. A la salida del inversor de 110 V_{AC} monofásico se conecta en serie el sensor de potencia y su salida se envía al pin D8 de la placa Arduino para ir contabilizando la energía consumida en AC por el sistema.

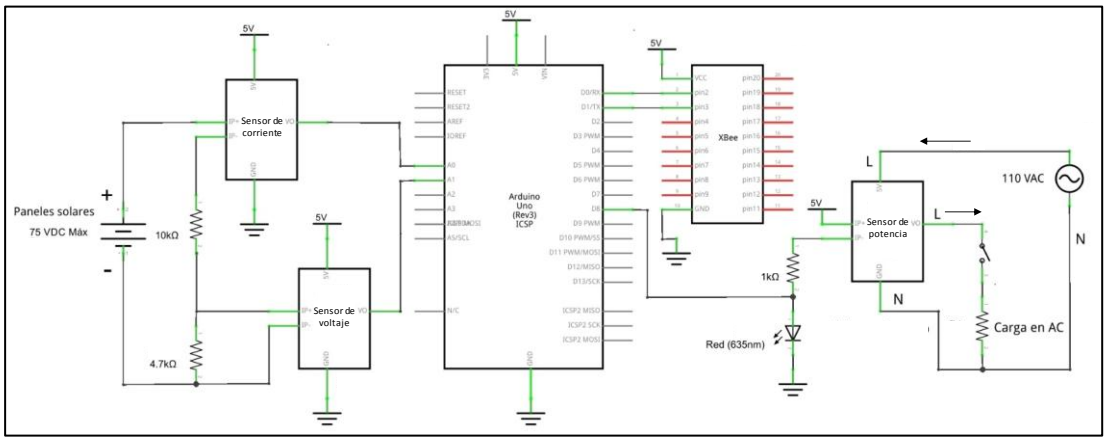

Figura 4.2. *Diseño del circuito de lectura de voltaje y corriente producidos y potencia consumida para la central fotovoltaica de 2.5 kW.*

- Se instaló un sensor de voltaje de 25 V_{DC} máximo y un sensor de corriente de 50 A máximo, además de un módulo *XBee* series 2.
- Se diseñó un circuito electrónico para la alimentación y envío de datos del dispositivo de radiofrecuencia, ver figura 4.1.1.
- Se instaló un medidor de energía consumida (kilowattimetro) para la lectura de energía consumida por el sistema en corriente alterna.
- Se elaboró un programa para la adquisición de los datos obtenidos por los sensores y el kilowattimetro en el software de programación de la placa Arduino.
- Se obtuvo un rango de cobertura de 85 m a campo abierto entre la central fotovoltaica y el centro de monitoreo.

4.3 Aerogenerador de 900 W

- Se diseñó un circuito electrónico para la lectura del voltaje y corriente producidos por el aerogenerador, así como para el consumo de potencia en corriente alterna, además de la alimentación de los sensores utilizados, como se puede observar en la figura 4.3.

El circuito de monitoreo se conecta en el controlador de carga, en la entrada proveniente del aerogenerador. El sensor de corriente se conecta en serie en el positivo de la entrada de dicho aerogenerador, y en paralelo se conecta el sensor de voltaje. Las señales de salida del sensor de corriente y voltaje van a las entradas analógicas A0 y A1 de la placa Arduino respectivamente.

Los pines 2 y 3 del módulo *XBee* se conectan al pin RX y TX de la placa Arduino respectivamente para el envío de los datos de manera inalámbrica. A la salida del inversor de 110 V_{AC} monofásico se conecta en serie el sensor de energía consumida y su salida se envía al pin D8 de la placa Arduino para ir contabilizando la energía consumida en AC por el sistema.

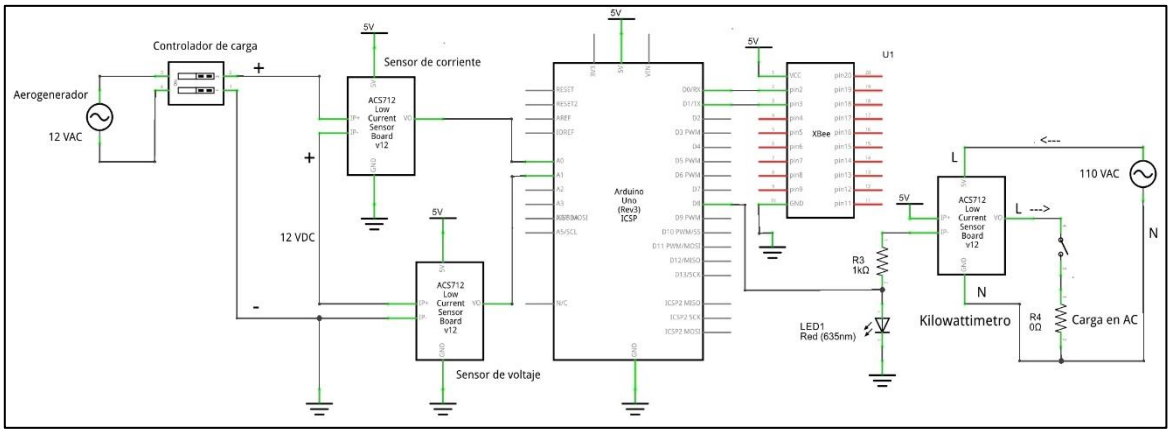

Figura 4.3. *Diseño del circuito de lectura de voltaje y corriente producidos y potencia consumida para el aerogenerador de 900 W.*

- Se instaló un sensor de voltaje de 25 V_{DC} máximo y un sensor de corriente de 100 A máximo, además de un módulo *XBee* serie 2.
- Se diseñó un circuito electrónico para la alimentación y envío de datos del dispositivo de radiofrecuencia.

- Se instaló un kilowattimetro para la lectura de potencia consumida por el sistema en AC.
- Se elaboró un programa para la adquisición de los datos obtenidos por los sensores y el kilowattimetro en el software de programación de la placa Arduino.
- Se obtuvo un rango de cobertura de 50 m a campo abierto entre el deshidratador y el centro de monitoreo.

4.4 Programa de interfaz del centro de monitoreo

- Se elaboró un algoritmo de programación para la adquisición y visualización de los datos del centro de monitoreo.

Se inicia el programa, busca los puertos COM (puerto serial de la computadora) disponibles, si no encuentra ningún puerto COM disponible, envía un mensaje de error y cierra el programa, si detecta uno o más puertos disponibles los visualiza y permite elegir el puerto con el cual se va a trabajar; si el puerto seleccionado no es correcto o no funciona, regresa a la etapa de buscar puertos disponibles; si el puerto es correcto, muestra un mensaje de “conexión lista”, y luego inicia el monitoreo, revisa la conexión a la base de datos, si no hay conexión a la base de datos causa un error, cierra el programa e indica que hay que iniciar el programa que establece la conexión entre *Microsoft Visual C# 2010 Express* y la base de datos, dicho programa es *WampServer64*; si la conexión se establece con éxito, envía un mensaje de “conectado” y realiza la conexión al servidor web para poder enviar la información a internet, si no hay conexión envía un error y se debe iniciar el servidor; luego comienza a leer el puerto serial para ver si recibe información, si no recibe datos continúa leyendo hasta que lee uno o más datos, cuando lee los datos, los separa de acuerdo a su codificación, los visualiza, los grafica, envía dicha información a la base de datos y al mismo tiempo envía la información a la página web del centro de monitoreo, este ciclo lo realiza continuamente sin interrupción, si el usuario decide cerrar el programa, *Visual C#* cierra todas las conexiones y con ello termina el proceso de monitoreo.

- Se desarrolló un programa para la interfaz del centro de monitoreo en *Microsoft Visual C# 2010 Express* (ver figura 4.4) en el cual se puede observar los datos adquiridos en tiempo real de cada uno de los sistemas en cuestión, y al mismo tiempo va graficando los valores obtenidos.
- Los datos visualizados y graficados en tiempo real son la fecha, hora, temperatura y humedad de la cámara de secado, temperatura del agua del tanque de almacenamiento del deshidratador híbrido solar - eólico.
- Los datos visualizados y graficados en tiempo real para la central fotovoltaica de 2.5 kW son la fecha, hora, voltaje y corriente producidos en DC y energía consumida en AC.
- Los datos visualizados y graficados en tiempo real para el aerogenerador de 900 W son la fecha, hora, voltaje y corriente producidos en AC y potencia consumida en AC.
- La información se guarda en una base de datos realizada en *MySQL Workbench 5.2 CE*, de la cual se pueden obtener los datos por medio de un archivo CSV para analizarlo en cualquier software de análisis de datos, ver figura 4.4.3.
- La velocidad de transmisión de datos de los dispositivos de radiofrecuencia es de 250 kbps.
- La frecuencia de trabajo del sistema de monitoreo es de 2.4 GHz.

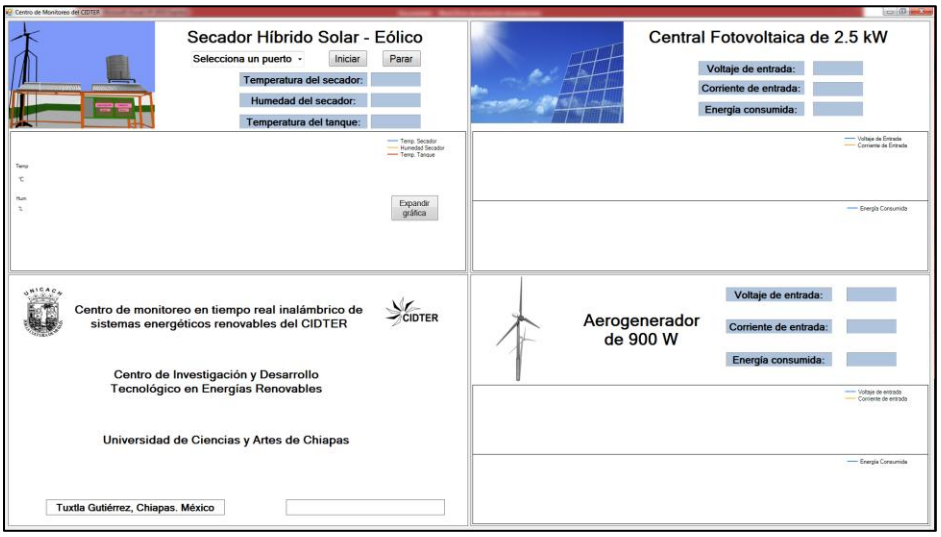

Figura 4.4. *Pantalla de visualización del centro de monitoreo.*

Como se puede observar en la figura 4.4, la pantalla del centro de monitoreo está dividida en 4 partes, en el superior izquierdo se visualizan los datos obtenidos del deshidratador híbrido solar – eólico como lo son la temperatura del secador, la humedad del secador y la temperatura del tanque de almacenamiento, así como la gráfica de dichos valores, además del *combobox* de la selección del puerto para iniciar el monitoreo y los botones de inicio y parar.

En el cuadro superior derecho está destinado para la central fotovoltaica autónoma de 2.5 kW y al igual que el anterior, se visualizan y grafican los datos obtenidos (voltaje y corriente de entrada de los paneles y energía consumida en AC). El cuadro inferior izquierdo es la parte de la presentación de los datos de la universidad, del centro de investigación, así como el lugar, la fecha y hora. El cuadro inferior derecho es para el aerogenerador de 900 W, en el cual se visualizan y grafican los datos obtenidos (voltaje y corriente de entrada del aerogenerador y energía consumida en AC).

En la figura 4.5 se observa el entorno del software *MySQL Workbench 5.2 CE*, en el cual se realizó la base de datos la cual se llama “historial”, la cual contiene 3 tablas, una para cada sistema monitoreado. Los nombres de las tablas son aerogenerador, central y secador (para el deshidratador híbrido solar – eólico).

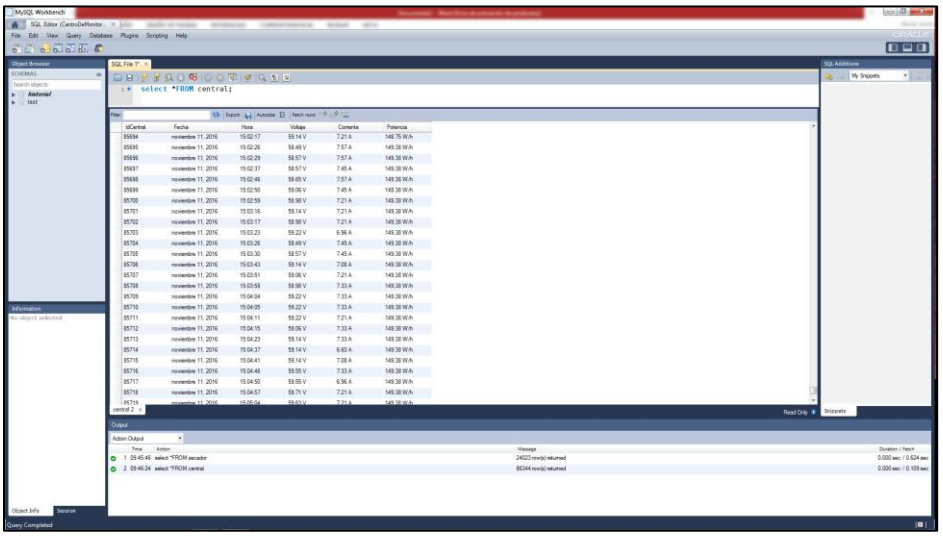

Figura 4.5. *Base de datos para el deshidratador híbrido solar-eólico.*

Como se puede observar en la figura 4.5, se realizó una consulta a la base de datos para poder visualizar la tabla de los registros del secador, el programa despliega una tabla con la fecha, la hora, y los valores obtenidos en ese momento del sistema, para este caso despliega los valores de la temperatura y humedad de la cámara de secado, así como la temperatura del agua del tanque de almacenamiento.

Los valores almacenados para la central fotovoltaica y el aerogenerador son: fecha, hora, voltaje y corriente de entrada en corriente directa, así como la energía consumida en corriente alterna.

4.5 Página web

- Se diseñó la página web del centro de monitoreo mediante la cual se puede observar en tiempo real desde cualquier equipo de cómputo, Tablet o Smartphone en cualquier lugar del mundo que tenga acceso a internet el comportamiento de cada uno de los sistemas monitoreados.

Una vez encendido el equipo de cómputo destinado como servidor, y desde el momento en el que se inicia el servidor virtual del centro de monitoreo (*WampServer64*) se habilita el modo “online” para activar los servicios de *Apache, PHP y MySQL* y con ello se puede acceder de manera remota al archivo *index.php* el cual contiene la página web del centro de monitoreo. El programa se realizó en el software llamado *Adobe Dreamweaver*.

Al iniciar el servidor virtual, el archivo que contiene a la página web llamado *index.php* realiza una conexión a la base de datos de *MySQL* mediante un archivo llamado *local.php* y se encarga constantemente de estar solicitando información a la base de datos mediante diferentes archivos que son exclusivos para cada una de los parámetros monitoreados como lo son temperatura, humedad, temperatura del tanque de almacenamiento, voltaje y corriente de entrada y energía consumida, así como la fecha y hora. Una vez que recibe la conexión de la base de datos y la información necesaria, el programa separa los valores de cada sistema para visualizarlos y al mismo tiempo irlos graficando, como se puede observar en la figura 4.6.

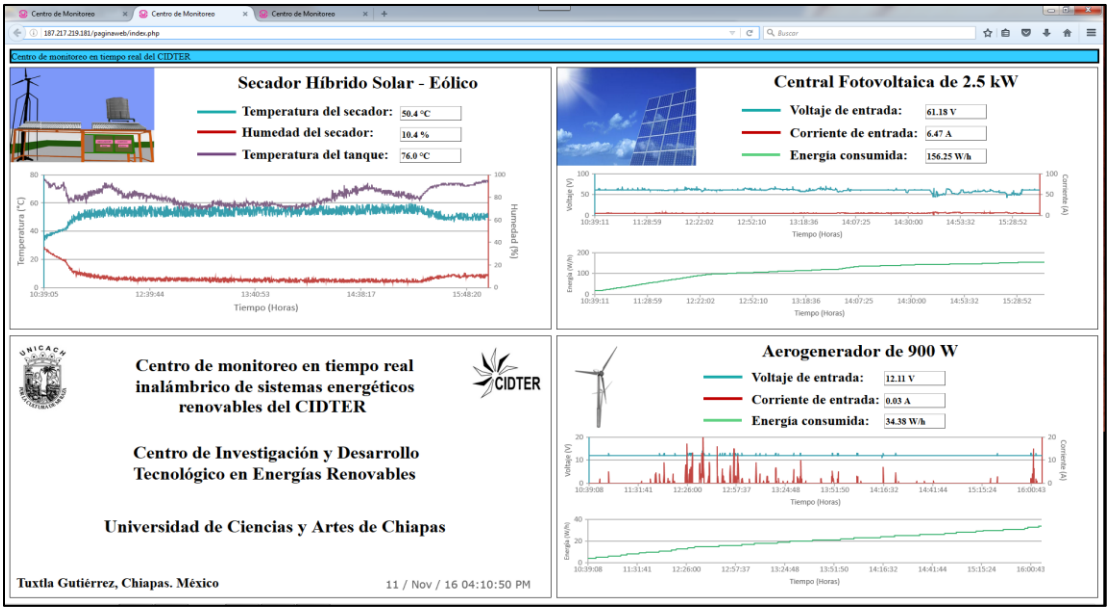

Figura 4.6. *Página web funcionando en tiempo real.*

4.6 Conclusiones

- Se logró monitorear 3 diferentes sistemas energéticos renovables, un aerogenerador de 900 W, una central fotovoltaica de 2.5 kW y un deshidratador híbrido solar-eólico.
- Se elaboró un programa de adquisición y visualización del centro de monitoreo.
- Se creó una base de datos de los sistemas monitoreados.
- Se diseñó una página web para el centro de monitoreo.
- Se alcanzó un rango de cobertura de 100 m entre los sistemas y el centro de monitoreo.
- Se terminaron satisfactoriamente cada uno los objetivos del proyecto.
- Se logró tener un monitoreo remoto por internet confiable.
- Actualmente se encuentra funcionando correctamente el centro de monitoreo en tiempo real.

Capítulo 5. Conclusiones generales

- Con el capítulo 1 tenemos una introducción al funcionamiento y los diferentes tipos de protocolos de comunicación de datos de manera inalámbrica.
- Se determinaron e instalaron los equipos necesarios en los tres diferentes sistemas energéticos renovables para el envío de datos de manera inalámbrica.
- La tecnología *ZigBee* es ideal para realizar un sistema de monitoreo inalámbrico gracias a que es configurable, tiene un mayor rango de alcance y se pueden realizar topologías en malla, y el *Bluetooth* y *Wi-Fi* no pueden realizar este tipo de configuraciones.
- Se logró monitorear 3 diferentes sistemas energéticos renovables, un aerogenerador de 900 W, una central fotovoltaica autónoma de 2.5 kW y un deshidratador híbrido solar-eólico.

- Los parámetros monitoreados son el voltaje y corriente de entrada en corriente directa y la energía consumida en corriente alterna para la central fotovoltaica autónoma y el aerogenerador.
- Los parámetros monitoreados en el deshidratador híbrido son la temperatura y humedad de la cámara de secado y la temperatura del agua en el tanque de almacenamiento.
- Se creó una base de datos de los sistemas monitoreados.
- Se logró un monitoreo remoto por internet mediante una página web.
- La frecuencia de trabajo del sistema es de 2.4 GHz.
- La velocidad de transmisión de datos alcanzada es de 250 kbps.

Trabajos futuros

- Instalar más sensores a cada sistema monitoreado (por ejemplo: anemómetros, piranómetros, sensores de cargar y descarga de baterías, sensores de vibración, entre otros) para tener un sistema de monitoreo más completo para cada uno de ellos.
- Realizar análisis detallado de los datos obtenidos por el centro de monitoreo (por ejemplo: análisis de cinética de secado en el caso del deshidratador híbrido; o análisis de potencia suministrada por la central fotovoltaica).
- Integrar un sistema de envío de mensajes de texto al usuario cuando exista fallo en algún sistema.
- Controlar los sistemas monitoreados de manera remota por medio de internet.

Anexo A

- **Código de Arduino para la adquisición de datos del Aerogenerador.**

```
#include <SoftwareSerial.h>

int i;
int entrada;
float lectura;
float lectura2;
float voltaje;
float corriente;
float potencia;

void setup() {
  Serial.begin(9600);
  pinMode(22, INPUT);
}

void loop()
{
  for(i=0; i < 400; i++)
  {
    lectura = analogRead(A8);
    voltaje = (lectura/40.8);

    lectura2 = analogRead(A9);
    corriente = ((lectura2-510)/17.05)-0.08797654;

    if(digitalRead(22) == HIGH)
  {
      entrada++;
      potencia = entrada*0.625;
      delay(100);
    }
    delay(10);
}

  Serial.print("AV.");
  Serial.print(voltaje);
  Serial.println(" V");

  if (corriente < 0)
      {
        corriente = (corriente)*(-1);
        Serial.print("AC.");
        Serial.print(corriente);
        Serial.println(" A");
        }
        else
        {
          Serial.print("AC.");
          Serial.print(corriente);
          Serial.println(" A");
        }
```

```
  Serial.print("AP.");
  Serial.print(potencia);
  Serial.println(" W");
}
```

- **Código de Arduino para la adquisición de datos de la Central fotovoltaica.**

```
#include <SoftwareSerial.h>

int i;
int entrada;
float potencia;
float lectura0;
float corriente;
float lectura1;
float voltaje;
float division;

void setup()
{
  Serial.begin(9600);
  pinMode(8, INPUT);
}

void loop()
{
for (i=0; i<400;i++)
{
    if(digitalRead(8) == HIGH)
  {
      entrada++;
      potencia = (entrada/1.6);
      delay(100);
    }
    delay(10);
  }

       lectura1 = analogRead(A1);
       division = lectura1/40.4;
       voltaje = division*3.3;
       Serial.print("CV.");
       Serial.print(voltaje);
       Serial.println(" V");

      lectura0 = analogRead(A0);
      corriente = (510 - lectura0)*5/1024/0.04 - 0.24;

      if (corriente < 0)
      {
        corriente = (corriente)*(-1);
```

```
      Serial.print("CC.");
      Serial.print(corriente);
      Serial.println(" A");
      }
      else
      {
        Serial.print("CC.");
        Serial.print(corriente);
        Serial.println(" A");
      }
    Serial.print("CP.");
    Serial.print(potencia);
    Serial.println(" W");
}
```

- **Código de Arduino para la adquisición de datos del Deshidratador.**

```
  Serial.print("ST.");
  Serial.print(temperature, 1);
  Serial.println(".C");
  //Serial.print("°C\n");
  Serial.print("SH.");
  Serial.print(humidity, 1);
  Serial.println(" %");
  //Serial.print("%\n");
  Serial.print("STT.");
  Serial.print(temptanq, 1);
  Serial.println(".C");
```

NOTA: Únicamente se modificó una parte del código del sistema de control del deshidratador. Por lo que no se escribió el código completo en este trabajo.

Anexo B

- **Código del programa de adquisición de datos del centro de monitoreo.**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.IO.Ports;
using System.Windows.Forms.DataVisualization.Charting;
using System.Windows.Forms;

namespace Centro_de_Monitoreo
{
   public partial class Principal : Form
   {
      public Principal()
      {
         InitializeComponent();
         Control.CheckForIllegalCrossThreadCalls = false;
         foreach (string buscar_puertos in SerialPort.GetPortNames())
         {
            comboBox1.Items.Add(buscar_puertos);
         }
      }

      private void Principal_Load(object sender, EventArgs e)
      {
         reinicio = 0;
         tiempo = 0;
         limpiar = 0;
      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         serialPort1.PortName = comboBox1.Text;
         try
         {
            MessageBox.Show("conexión lista!", "listo", MessageBoxButtons.OK);
         }

         catch
         {
            MessageBox.Show("El puerto seleccionado no envia ni recibe ningun dato. Seleccione otro puerto.",
"Error puerto", MessageBoxButtons.OK);
            return;
         }
         comboBox1.Enabled = false;
       }

      private void conectar_Click(object sender, EventArgs e)
      {
         conectar.Enabled = false;
         Desconectar.Enabled = true;
         timer1.Enabled = true;
         timer2.Enabled = true;
         timer3.Enabled = true;
```

```
    serialPort1.Open();
    BdComun.ObtenerConexion();
    MessageBox.Show("conectado");
}

private void Desconectar_Click(object sender, EventArgs e)
{
    Desconectar.Enabled = false;
    conectar.Enabled = true;
    timer1.Enabled = false;
    timer2.Enabled = false;
    serialPort1.Close();
}

private void timer1_Tick(object sender, EventArgs e)
{
    Proceso();
    reinicio++;
    if (reinicio == 1)
    {
        reinicio = 0;
        timer1.Dispose();
        timer1.Start();
    }

}

private void timer2_Tick(object sender, EventArgs e)
{
    tiempo++;
    limpiar++;
    if (tiempo == 100)
    {
        tiempo = 0;
        timer2.Dispose();
        timer2.Start();
    }

    if (limpiar == 6000)
    {
        chart1.Series[0].Points.Clear();
        chart1.Series[1].Points.Clear();
        chart1.Series[2].Points.Clear();
        chart2.Series[0].Points.Clear();
        chart2.Series[1].Points.Clear();
        chart3.Series[0].Points.Clear();
        chart4.Series[0].Points.Clear();
        chart4.Series[1].Points.Clear();
        chart5.Series[0].Points.Clear();

        lblTemp.Text = " ";
        lblHum.Text = " ";
        lblTanq.Text = " ";
        Voltajecentral.Text = " ";
        CorrienteCentral.Text = " ";
        kWConsumidos.Text = " ";
        VoltajeAero.Text = " ";
        CorrienteAero.Text = " ";
        kWconsumidosAero.Text = " ";


        limpiar = 0;
```

```
    }

}

private void timer3_Tick(object sender, EventArgs e)
{
    Fecha.Text = DateTime.Now.ToString("MMMM dd, yyyy    HH:mm:ss");

}

public int reinicio;
public int tiempo;
public int limpiar;
public string bdsTemperatura;
public string bdsHumedad;
public string bdsTempTanque;
public string bdcVoltaje;
public string bdcCorriente;
public string bdcPotencia;
public string bdaVoltaje;
public string bdaCorriente;
public string bdaPotencia;
public string a;
public string b;
public string c;
public string d;
public string e;
public string f;
public string g;
public string h;
public string i;
public string j;
public string k;
public string m;
public string aa;
public string bb;
public string cc;
public string dd;
public string ee;
public string ff;
public string gg;
public string hh;
public string ii;
public string jj;
public string kk;
public string mm;
public string Tempsecador;
public string Humedad;
public string temptanque;
public string Volcentral;
public string Corcentral;
public string Potcentral;
public string Volaero;
public string Coraero;
public string Potaero;
int valorminimo, valormaximo;
public int TP, HP, TqP;
public int VC, CC, PC;
public int VA, CA, PA;
public void Proceso()
{
    try
```

```
{
    char[] limite = new char[] { '.' };
    a = serialPort1.ReadLine();
    string[] Temperatura = a.Split(limite);
    aa = Temperatura[0];
    if (aa == "ST")
    {
        lblTemp.Text = Temperatura[1] + "." + Temperatura[2] + " °" + Temperatura[3];
        Tempsecador = Temperatura[1];
        TP = int.Parse(Tempsecador);
        bdsTemperatura = Temperatura[1] + "." + Temperatura[2] + " " + Temperatura[3];
        chart1.Series[0].XValueType = ChartValueType.String;
        chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
        Secador pSecador = new Secador();
        pSecador.Fecha = label17.Text.Trim();
        pSecador.Hora = DateTime.Now.ToString("HH:mm:ss");
        pSecador.Temp_secador = bdsTemperatura;
        pSecador.Humedad_secador = bdsHumedad;
        pSecador.Temp_tanque = bdsTempTanque;
        int resultado = SecadorDAL.Agregar(pSecador);
    }
    if (aa == "SH")
    {
        lblHum.Text = Temperatura[1] + "." + Temperatura[2];
        Humedad = Temperatura[1];
        HP = int.Parse(Humedad);
        bdsHumedad = Temperatura[1] + "." + Temperatura[2];
        chart1.Series[1].XValueType = ChartValueType.String;
        chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
    }

    if (aa == "STT")
    {
        lblTanq.Text = Temperatura[1] + "." + Temperatura[2] + " °" + Temperatura[3];
        temptanque = Temperatura[1];
        TqP = int.Parse(temptanque);
        bdsTempTanque = Temperatura[1] + "." + Temperatura[2] + " " + Temperatura[3];
        chart1.Series[2].XValueType = ChartValueType.String;
        chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
    }

    if (aa == "CV")
    {
        Voltajecentral.Text = Temperatura[1] + "." + Temperatura[2];// +" " + Temperatura[3];
        Volcentral = Temperatura[1];
        VC = int.Parse(Volcentral);
        bdcVoltaje = Temperatura[1] + "." + Temperatura[2];
        chart2.Series[0].XValueType = ChartValueType.String;
        chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
        Central pCentral = new Central();
        pCentral.Fecha = label17.Text.Trim();
        pCentral.Hora = DateTime.Now.ToString("HH:mm:ss");
        pCentral.Voltaje = bdcVoltaje;
        pCentral.Corriente = bdcCorriente;
        pCentral.Potencia = bdcPotencia;
        int resultado = CentralDAL.Agregar(pCentral);
    }

    if (aa == "CC")
    {
        CorrienteCentral.Text = Temperatura[1] + "." + Temperatura[2];// +" " + Temperatura[3];
        Corcentral = Temperatura[1];
```

```
    CC = int.Parse(Corcentral);
    bdcCorriente = Temperatura[1] + "." + Temperatura[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (aa == "CP")
{
    kWConsumidos.Text = Temperatura[1] + "." + Temperatura[2];// +" " + Temperatura[3];
    Potcentral = Temperatura[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = Temperatura[1] + "." + Temperatura[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (aa == "AV")
{
    VoltajeAero.Text = Temperatura[1] + "." + Temperatura[2];// +" " + Temperatura[3];
    Volaero = Temperatura[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = Temperatura[1] + "." + Temperatura[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (aa == "AC")
{
    CorrienteAero.Text = Temperatura[1] + "." + Temperatura[2];// +" " + Temperatura[3];
    Coraero = Temperatura[1];
    CA = int.Parse(Coraero);
    bdaCorriente = Temperatura[1] + "." + Temperatura[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
}

if (aa == "AP")
{
    kWconsumidosAero.Text = Temperatura[1] + "." + Temperatura[2];// +" " + Temperatura[3];
    Potaero = Temperatura[1];
    PA = int.Parse(Potaero);
    bdaPotencia = Temperatura[1] + "." + Temperatura[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}


b = serialPort1.ReadLine();
string[] Humidity = b.Split(limite);
bb = Humidity[0];
if (bb == "ST")
{
    lblTemp.Text = Humidity[1] + "." + Humidity[2] + " °" + Humidity[3];
    Tempsecador = Humidity[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = Humidity[1] + "." + Humidity[2] + " " + Humidity[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
}

if (bb == "SH")
```

```
{
    lblHum.Text = Humidity[1] + "." + Humidity[2];
    Humedad = Humidity[1];
    HP = int.Parse(Humedad);
    bdsHumedad = Humidity[1] + "." + Humidity[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (bb == "STT")
{
    lblTanq.Text = Humidity[1] + "." + Humidity[2] + " °" + Humidity[3];
    temptanque = Humidity[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = Humidity[1] + "." + Humidity[2] + " " + Humidity[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (bb == "CV")
{
    Voltajecentral.Text = Humidity[1] + "." + Humidity[2];// +" " + Humidity[3];
    Volcentral = Humidity[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = Humidity[1] + "." + Humidity[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
}

if (bb == "CC")
{
    CorrienteCentral.Text = Humidity[1] + "." + Humidity[2];// +" " + Humidity[3];
    Corcentral = Humidity[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = Humidity[1] + "." + Humidity[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (bb == "CP")
{
    kWConsumidos.Text = Humidity[1] + "." + Humidity[2];// +" " + Humidity[3];
    Potcentral = Humidity[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = Humidity[1] + "." + Humidity[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (bb == "AV")
{
    VoltajeAero.Text = Humidity[1] + "." + Humidity[2];// +" " + Humidity[3];
    Volaero = Humidity[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = Humidity[1] + "." + Humidity[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (bb == "AC")
{
```

```
    CorrienteAero.Text = Humidity[1] + "." + Humidity[2];// +" " + Humidity[3];
    Coraero = Humidity[1];
    CA = int.Parse(Coraero);
    bdaCorriente = Humidity[1] + "." + Humidity[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
    Aerogenerador pAerogenerador = new Aerogenerador();
    pAerogenerador.Fecha = label17.Text.Trim();
    pAerogenerador.Hora = DateTime.Now.ToString("HH:mm:ss");
    pAerogenerador.Voltaje = bdaVoltaje;
    pAerogenerador.Corriente = bdaCorriente;
    pAerogenerador.Potencia = bdaPotencia;
    int resultado = AerogeneradorDAL.Agregar(pAerogenerador);
}

if (bb == "AP")
{
    kWconsumidosAero.Text = Humidity[1] + "." + Humidity[2];// +" " + Humidity[3];
    Potaero = Humidity[1];
    PA = int.Parse(Potaero);
    bdaPotencia = Humidity[1] + "." + Humidity[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}


c = serialPort1.ReadLine();
string[] TempTanq = c.Split(limite);
cc = TempTanq[0];
if (cc == "ST")
{
    lblTemp.Text = TempTanq[1] + "." + TempTanq[2] + " °" + TempTanq[3];
    Tempsecador = TempTanq[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = TempTanq[1] + "." + TempTanq[2] + " " + TempTanq[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
}

if (cc == "SH")
{
    lblHum.Text = TempTanq[1] + "." + TempTanq[2];
    Humedad = TempTanq[1];
    HP = int.Parse(Humedad);
    bdsHumedad = TempTanq[1] + "." + TempTanq[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (cc == "STT")
{
    lblTanq.Text = TempTanq[1] + "." + TempTanq[2] + " °" + TempTanq[3];
    temptanque = TempTanq[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = TempTanq[1] + "." + TempTanq[2] + " " + TempTanq[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (cc == "CV")
{
    Voltajecentral.Text = TempTanq[1] + "." + TempTanq[2];// +" " + TempTanq[3];
```

```
    Volcentral = TempTanq[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = TempTanq[1] + "." + TempTanq[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
}

if (cc == "CC")
{
    CorrienteCentral.Text = TempTanq[1] + "." + TempTanq[2];// +" " + TempTanq[3];
    Corcentral = TempTanq[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = TempTanq[1] + "." + TempTanq[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (cc == "CP")
{
    kWConsumidos.Text = TempTanq[1] + "." + TempTanq[2];// +" " + TempTanq[3];
    Potcentral = TempTanq[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = TempTanq[1] + "." + TempTanq[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (cc == "AV")
{
    VoltajeAero.Text = TempTanq[1] + "." + TempTanq[2];// +" " + TempTanq[3];
    Volaero = TempTanq[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = TempTanq[1] + "." + TempTanq[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (cc == "AC")
{
    CorrienteAero.Text = TempTanq[1] + "." + TempTanq[2];// +" " + TempTanq[3];
    Coraero = TempTanq[1];
    CA = int.Parse(Coraero);
    bdaCorriente = TempTanq[1] + "." + TempTanq[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
}

if (cc == "AP")
{
    kWconsumidosAero.Text = TempTanq[1] + "." + TempTanq[2];// +" " + TempTanq[3];
    Potaero = TempTanq[1];
    PA = int.Parse(Potaero);
    bdaPotencia = TempTanq[1] + "." + TempTanq[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}


d = serialPort1.ReadLine();
string[] VolEnCentral = d.Split(limite);
dd = VolEnCentral[0];
```

```
if (dd == "ST")
{
    lblTemp.Text = VolEnCentral[1] + "." + VolEnCentral[2] + " °" + VolEnCentral[3];
    Tempsecador = VolEnCentral[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = VolEnCentral[1] + "." + VolEnCentral[2] + " " + VolEnCentral[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
    Secador pSecador = new Secador();
    pSecador.Fecha = label17.Text.Trim();
    pSecador.Hora = DateTime.Now.ToString("HH:mm:ss");
    pSecador.Temp_secador = bdsTemperatura;
    pSecador.Humedad_secador = bdsHumedad;
    pSecador.Temp_tanque = bdsTempTanque;
    int resultado = SecadorDAL.Agregar(pSecador);
}

if (dd == "SH")
{
    lblHum.Text = VolEnCentral[1] + "." + VolEnCentral[2];
    Humedad = VolEnCentral[1];
    HP = int.Parse(Humedad);
    bdsHumedad = VolEnCentral[1] + "." + VolEnCentral[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (dd == "STT")
{
    lblTanq.Text = VolEnCentral[1] + "." + VolEnCentral[2] + " °" + VolEnCentral[3];
    temptanque = VolEnCentral[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = VolEnCentral[1] + "." + VolEnCentral[2] + " " + VolEnCentral[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (dd == "CV")
{
    Voltajecentral.Text = VolEnCentral[1] + "." + VolEnCentral[2];// +" " + VolEnCentral[3];
    Volcentral = VolEnCentral[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = VolEnCentral[1] + "." + VolEnCentral[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
    Central pCentral = new Central();
    pCentral.Fecha = label17.Text.Trim();
    pCentral.Hora = DateTime.Now.ToString("HH:mm:ss");
    pCentral.Voltaje = bdcVoltaje;
    pCentral.Corriente = bdcCorriente;
    pCentral.Potencia = bdcPotencia;
    int resultado = CentralDAL.Agregar(pCentral);
}

if (dd == "CC")
{
    CorrienteCentral.Text = VolEnCentral[1] + "." + VolEnCentral[2];// +" " + VolEnCentral[3];
    Corcentral = VolEnCentral[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = VolEnCentral[1] + "." + VolEnCentral[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
```

```
}

if (dd == "CP")
{
    kWConsumidos.Text = VolEnCentral[1] + "." + VolEnCentral[2];// +" " + VolEnCentral[3];
    Potcentral = VolEnCentral[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = VolEnCentral[1] + "." + VolEnCentral[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (dd == "AV")
{
    VoltajeAero.Text = VolEnCentral[1] + "." + VolEnCentral[2];// +" " + VolEnCentral[3];
    Volaero = VolEnCentral[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = VolEnCentral[1] + "." + VolEnCentral[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (dd == "AC")
{
    CorrienteAero.Text = VolEnCentral[1] + "." + VolEnCentral[2];// +" " + VolEnCentral[3];
    Coraero = VolEnCentral[1];
    CA = int.Parse(Coraero);
    bdaCorriente = VolEnCentral[1] + "." + VolEnCentral[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
}

if (dd == "AP")
{
    kWconsumidosAero.Text = VolEnCentral[1] + "." + VolEnCentral[2];// +" " + VolEnCentral[3];
    Potaero = VolEnCentral[1];
    PA = int.Parse(Potaero);
    bdaPotencia = VolEnCentral[1] + "." + VolEnCentral[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}


e = serialPort1.ReadLine();
string[] CorEnCentral = e.Split(limite);
ee = CorEnCentral[0];
if (ee == "ST")
{
    lblTemp.Text = CorEnCentral[1] + "." + CorEnCentral[2] + " °" + CorEnCentral[3];
    Tempsecador = CorEnCentral[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = CorEnCentral[1] + "." + CorEnCentral[2] + " " + CorEnCentral[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
}

if (ee == "SH")
{
    lblHum.Text = CorEnCentral[1] + "." + CorEnCentral[2];
    Humedad = CorEnCentral[1];
    HP = int.Parse(Humedad);
```

```
    bdsHumedad = CorEnCentral[1] + "." + CorEnCentral[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (ee == "STT")
{
    lblTanq.Text = CorEnCentral[1] + "." + CorEnCentral[2] + " °" + CorEnCentral[3];
    temptanque = CorEnCentral[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = CorEnCentral[1] + "." + CorEnCentral[2] + " " + CorEnCentral[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (ee == "CV")
{
    Voltajecentral.Text = CorEnCentral[1] + "." + CorEnCentral[2];// +" " + CorEnCentral[3];
    Volcentral = CorEnCentral[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = CorEnCentral[1] + "." + CorEnCentral[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
 }

if (ee == "CC")
{
    CorrienteCentral.Text = CorEnCentral[1] + "." + CorEnCentral[2];// +" " + CorEnCentral[3];
    Corcentral = CorEnCentral[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = CorEnCentral[1] + "." + CorEnCentral[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (ee == "CP")
{
    kWConsumidos.Text = CorEnCentral[1] + "." + CorEnCentral[2];// +" " + CorEnCentral[3];
    Potcentral = CorEnCentral[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = CorEnCentral[1] + "." + CorEnCentral[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (ee == "AV")
{
    VoltajeAero.Text = CorEnCentral[1] + "." + CorEnCentral[2];// +" " + CorEnCentral[3];
    Volaero = CorEnCentral[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = CorEnCentral[1] + "." + CorEnCentral[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (ee == "AC")
{
    CorrienteAero.Text = CorEnCentral[1] + "." + CorEnCentral[2];// +" " + CorEnCentral[3];
    Coraero = CorEnCentral[1];
    CA = int.Parse(Coraero);
    bdaCorriente = CorEnCentral[1] + "." + CorEnCentral[2];
```

```
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
    Aerogenerador pAerogenerador = new Aerogenerador();
    pAerogenerador.Fecha = label17.Text.Trim();
    pAerogenerador.Hora = DateTime.Now.ToString("HH:mm:ss");
    pAerogenerador.Voltaje = bdaVoltaje;
    pAerogenerador.Corriente = bdaCorriente;
    pAerogenerador.Potencia = bdaPotencia;
    int resultado = AerogeneradorDAL.Agregar(pAerogenerador);
}

if (ee == "AP")
{
    kWconsumidosAero.Text = CorEnCentral[1] + "." + CorEnCentral[2];// +" " + CorEnCentral[3];
    Potaero = CorEnCentral[1];
    PA = int.Parse(Potaero);
    bdaPotencia = CorEnCentral[1] + "." + CorEnCentral[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}


f = serialPort1.ReadLine();
string[] PotSaCentral = f.Split(limite);
ff = PotSaCentral[0];
if (ff == "ST")
{
    lblTemp.Text = PotSaCentral[1] + "." + PotSaCentral[2] + " °" + PotSaCentral[3];
    Tempsecador = PotSaCentral[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = PotSaCentral[1] + "." + PotSaCentral[2] + " " + PotSaCentral[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
}

if (ff == "SH")
{
    lblHum.Text = PotSaCentral[1] + "." + PotSaCentral[2];
    Humedad = PotSaCentral[1];
    HP = int.Parse(Humedad);
    bdsHumedad = PotSaCentral[1] + "." + PotSaCentral[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (ff == "STT")
{
    lblTanq.Text = PotSaCentral[1] + "." + PotSaCentral[2] + " °" + PotSaCentral[3];
    temptanque = PotSaCentral[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = PotSaCentral[1] + "." + PotSaCentral[2] + " " + PotSaCentral[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (ff == "CV")
{
    Voltajecentral.Text = PotSaCentral[1] + "." + PotSaCentral[2];// +" " + PotSaCentral[3];
    Volcentral = PotSaCentral[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = PotSaCentral[1] + "." + PotSaCentral[2];
    chart2.Series[0].XValueType = ChartValueType.String;
```

```
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
}

if (ff == "CC")
{
    CorrienteCentral.Text = PotSaCentral[1] + "." + PotSaCentral[2];// +" " + PotSaCentral[3];
    Corcentral = PotSaCentral[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = PotSaCentral[1] + "." + PotSaCentral[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (ff == "CP")
{
    kWConsumidos.Text = PotSaCentral[1] + "." + PotSaCentral[2];// +" " + PotSaCentral[3];
    Potcentral = PotSaCentral[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = PotSaCentral[1] + "." + PotSaCentral[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}

if (ff == "AV")
{
    VoltajeAero.Text = PotSaCentral[1] + "." + PotSaCentral[2];// +" " + PotSaCentral[3];
    Volaero = PotSaCentral[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = PotSaCentral[1] + "." + PotSaCentral[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (ff == "AC")
{
    CorrienteAero.Text = PotSaCentral[1] + "." + PotSaCentral[2];// +" " + PotSaCentral[3];
    Coraero = PotSaCentral[1];
    CA = int.Parse(Coraero);
    bdaCorriente = PotSaCentral[1] + "." + PotSaCentral[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
}

if (ff == "AP")
{
    kWconsumidosAero.Text = PotSaCentral[1] + "." + PotSaCentral[2];// +" " + PotSaCentral[3];
    Potaero = PotSaCentral[1];
    PA = int.Parse(Potaero);
    bdaPotencia = PotSaCentral[1] + "." + PotSaCentral[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}


g = serialPort1.ReadLine();
string[] VolEnAero = g.Split(limite);
gg = VolEnAero[0];
if (gg == "ST")
{
    lblTemp.Text = VolEnAero[1] + "." + VolEnAero[2] + " °" + VolEnAero[3];
    Tempsecador = VolEnAero[1];
    TP = int.Parse(Tempsecador);
```

```
    bdsTemperatura = VolEnAero[1] + "." + VolEnAero[2] + " " + VolEnAero[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
    Secador pSecador = new Secador();
    pSecador.Fecha = label17.Text.Trim();
    pSecador.Hora = DateTime.Now.ToString("HH:mm:ss");
    pSecador.Temp_secador = bdsTemperatura;
    pSecador.Humedad_secador = bdsHumedad;
    pSecador.Temp_tanque = bdsTempTanque;
    int resultado = SecadorDAL.Agregar(pSecador);
}

if (gg == "SH")
{
    lblHum.Text = VolEnAero[1] + "." + VolEnAero[2];
    Humedad = VolEnAero[1];
    HP = int.Parse(Humedad);
    bdsHumedad = VolEnAero[1] + "." + VolEnAero[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (gg == "STT")
{
    lblTanq.Text = VolEnAero[1] + "." + VolEnAero[2] + " °" + VolEnAero[3];
    temptanque = VolEnAero[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = VolEnAero[1] + "." + VolEnAero[2] + " " + VolEnAero[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (gg == "CV")
{
    Voltajecentral.Text = VolEnAero[1] + "." + VolEnAero[2];// +" " + VolEnAero[3];
    Volcentral = VolEnAero[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = VolEnAero[1] + "." + VolEnAero[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
    Central pCentral = new Central();
    pCentral.Fecha = label17.Text.Trim();
    pCentral.Hora = DateTime.Now.ToString("HH:mm:ss");
    pCentral.Voltaje = bdcVoltaje;
    pCentral.Corriente = bdcCorriente;
    pCentral.Potencia = bdcPotencia;
    int resultado = CentralDAL.Agregar(pCentral);
}

if (gg == "CC")
{
    CorrienteCentral.Text = VolEnAero[1] + "." + VolEnAero[2];// +" " + VolEnAero[3];
    Corcentral = VolEnAero[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = VolEnAero[1] + "." + VolEnAero[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (gg == "CP")
{
    kWConsumidos.Text = VolEnAero[1] + "." + VolEnAero[2];// +" " + VolEnAero[3];
```

```
    Potcentral = VolEnAero[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = VolEnAero[1] + "." + VolEnAero[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (gg == "AV")
{
    VoltajeAero.Text = VolEnAero[1] + "." + VolEnAero[2];// +" " + VolEnAero[3];
    Volaero = VolEnAero[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = VolEnAero[1] + "." + VolEnAero[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (gg == "AC")
{
    CorrienteAero.Text = VolEnAero[1] + "." + VolEnAero[2];// +" " + VolEnAero[3];
    Coraero = VolEnAero[1];
    CA = int.Parse(Coraero);
    bdaCorriente = VolEnAero[1] + "." + VolEnAero[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
}

if (gg == "AP")
{
    kWconsumidosAero.Text = VolEnAero[1] + "." + VolEnAero[2];// +" " + VolEnAero[3];
    Potaero = VolEnAero[1];
    PA = int.Parse(Potaero);
    bdaPotencia = VolEnAero[1] + "." + VolEnAero[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}



h = serialPort1.ReadLine();
string[] CorEnAero = h.Split(limite);

hh = CorEnAero[0];
if (hh == "ST")
{
    lblTemp.Text = CorEnAero[1] + "." + CorEnAero[2] + " °" + CorEnAero[3];
    Tempsecador = CorEnAero[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = CorEnAero[1] + "." + CorEnAero[2] + " " + CorEnAero[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
}

if (hh == "SH")
{
    lblHum.Text = CorEnAero[1] + "." + CorEnAero[2];
    Humedad = CorEnAero[1];
    HP = int.Parse(Humedad);
    bdsHumedad = CorEnAero[1] + "." + CorEnAero[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
```

```
}

if (hh == "STT")
{
    lblTanq.Text = CorEnAero[1] + "." + CorEnAero[2] + " °" + CorEnAero[3];
    temptanque = CorEnAero[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = CorEnAero[1] + "." + CorEnAero[2] + " " + CorEnAero[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (hh == "CV")
{
    Voltajecentral.Text = CorEnAero[1] + "." + CorEnAero[2];// +" " + CorEnAero[3];
    Volcentral = CorEnAero[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = CorEnAero[1] + "." + CorEnAero[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
}

if (hh == "CC")
{
    CorrienteCentral.Text = CorEnAero[1] + "." + CorEnAero[2];// +" " + CorEnAero[3];
    Corcentral = CorEnAero[1];
    CC = int.Parse(Corcentral);
    bdcCorriente = CorEnAero[1] + "." + CorEnAero[2];
    chart2.Series[1].XValueType = ChartValueType.String;
    chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
}

if (hh == "CP")
{
    kWConsumidos.Text = CorEnAero[1] + "." + CorEnAero[2];// +" " + CorEnAero[3];
    Potcentral = CorEnAero[1];
    PC = int.Parse(Potcentral);
    bdcPotencia = CorEnAero[1] + "." + CorEnAero[2];
    chart3.Series[0].XValueType = ChartValueType.String;
    chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
}


if (hh == "AV")
{
    VoltajeAero.Text = CorEnAero[1] + "." + CorEnAero[2];// +" " + CorEnAero[3];
    Volaero = CorEnAero[1];
    VA = int.Parse(Volaero);
    bdaVoltaje = CorEnAero[1] + "." + CorEnAero[2];
    chart4.Series[0].XValueType = ChartValueType.String;
    chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
}

if (hh == "AC")
{
    CorrienteAero.Text = CorEnAero[1] + "." + CorEnAero[2];// +" " + CorEnAero[3];
    Coraero = CorEnAero[1];
    CA = int.Parse(Coraero);
    bdaCorriente = CorEnAero[1] + "." + CorEnAero[2];
    chart4.Series[1].XValueType = ChartValueType.String;
    chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
    Aerogenerador pAerogenerador = new Aerogenerador();
```

```
    pAerogenerador.Fecha = label17.Text.Trim();
    pAerogenerador.Hora = DateTime.Now.ToString("HH:mm:ss");
    pAerogenerador.Voltaje = bdaVoltaje;
    pAerogenerador.Corriente = bdaCorriente;
    pAerogenerador.Potencia = bdaPotencia;
    int resultado = AerogeneradorDAL.Agregar(pAerogenerador);
}

if (hh == "AP")
{
    kWconsumidosAero.Text = CorEnAero[1] + "." + CorEnAero[2];// +" " + CorEnAero[3];
    Potaero = CorEnAero[1];
    PA = int.Parse(Potaero);
    bdaPotencia = CorEnAero[1] + "." + CorEnAero[2];
    chart5.Series[0].XValueType = ChartValueType.String;
    chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
}



i = serialPort1.ReadLine();
string[] PotSaAero = i.Split(limite);
ii = PotSaAero[0];
if (ii == "ST")
{
    lblTemp.Text = PotSaAero[1] + "." + PotSaAero[2] + " °" + PotSaAero[3];
    Tempsecador = PotSaAero[1];
    TP = int.Parse(Tempsecador);
    bdsTemperatura = PotSaAero[1] + "." + PotSaAero[2] + " " + PotSaAero[3];
    chart1.Series[0].XValueType = ChartValueType.String;
    chart1.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TP);
}

if (ii == "SH")
{
    lblHum.Text = PotSaAero[1] + "." + PotSaAero[2];
    Humedad = PotSaAero[1];
    HP = int.Parse(Humedad);
    bdsHumedad = PotSaAero[1] + "." + PotSaAero[2];
    chart1.Series[1].XValueType = ChartValueType.String;
    chart1.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), HP);
}

if (ii == "STT")
{
    lblTanq.Text = PotSaAero[1] + "." + PotSaAero[2] + " °" + PotSaAero[3];
    temptanque = PotSaAero[1];
    TqP = int.Parse(temptanque);
    bdsTempTanque = PotSaAero[1] + "." + PotSaAero[2] + " " + PotSaAero[3];
    chart1.Series[2].XValueType = ChartValueType.String;
    chart1.Series[2].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), TqP);
}

if (ii == "CV")
{
    Voltajecentral.Text = PotSaAero[1] + "." + PotSaAero[2];// +" " + PotSaAero[3];
    Volcentral = PotSaAero[1];
    VC = int.Parse(Volcentral);
    bdcVoltaje = PotSaAero[1] + "." + PotSaAero[2];
    chart2.Series[0].XValueType = ChartValueType.String;
    chart2.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VC);
}
```

```
        if (ii == "CC")
        {
            CorrienteCentral.Text = PotSaAero[1] + "." + PotSaAero[2];// +" " + PotSaAero[3];
            Corcentral = PotSaAero[1];
            CC = int.Parse(Corcentral);
            bdcCorriente = PotSaAero[1] + "." + PotSaAero[2];
            chart2.Series[1].XValueType = ChartValueType.String;
            chart2.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CC);
        }

        if (ii == "CP")
        {
            kWConsumidos.Text = PotSaAero[1] + "." + PotSaAero[2];// +" " + PotSaAero[3];
            Potcentral = PotSaAero[1];
            PC = int.Parse(Potcentral);
            bdcPotencia = PotSaAero[1] + "." + PotSaAero[2];
            chart3.Series[0].XValueType = ChartValueType.String;
            chart3.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PC);
        }

        if (ii == "AV")
        {
            VoltajeAero.Text = PotSaAero[1] + "." + PotSaAero[2];// +" " + PotSaAero[3];
            Volaero = PotSaAero[1];
            VA = int.Parse(Volaero);
            bdaVoltaje = PotSaAero[1] + "." + PotSaAero[2];
            chart4.Series[0].XValueType = ChartValueType.String;
            chart4.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), VA);
        }

        if (ii == "AC")
        {
            CorrienteAero.Text = PotSaAero[1] + "." + PotSaAero[2];// +" " + PotSaAero[3];
            Coraero = PotSaAero[1];
            CA = int.Parse(Coraero);
            bdaCorriente = PotSaAero[1] + "." + PotSaAero[2];
            chart4.Series[1].XValueType = ChartValueType.String;
            chart4.Series[1].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), CA);
        }

        if (ii == "AP")
        {
            kWconsumidosAero.Text = PotSaAero[1] + "." + PotSaAero[2];// +" " + PotSaAero[3];
            Potaero = PotSaAero[1];
            PA = int.Parse(Potaero);
            bdaPotencia = PotSaAero[1] + "." + PotSaAero[2];
            chart5.Series[0].XValueType = ChartValueType.String;
            chart5.Series[0].Points.AddXY(DateTime.Now.ToString("HH:mm:ss"), PA);
        }

    }

    catch (Exception)
    {

    }
}

private void Secador_Click(object sender, EventArgs e)
{
    serialPort1.Close();
```

```
            this.Desconectar.Enabled = false;
            this.conectar.Enabled = true;
            Grafica abre_ventana_secador = new Grafica();
            abre_ventana_secador.Show();
        }


        private void Principal_FormClosed(object sender, FormClosedEventArgs e)
        {
            timer1.Enabled = false;
            timer2.Enabled = false;
            timer3.Enabled = false;
            serialPort1.Close();
        }

    }
}
```

Anexo C

- **Código de la página web del centro de monitoreo.**

```
<?php require_once('Connections/local.php');
?>
<?php
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Centro de Monitoreo</title>
<style type="text/css">
#base {
	height: 990px;
	width: 1905px;
	margin-top: auto;
	margin-bottom: auto;
}
#base #contenido {
	background-color: #69F;
	width: 1000px;
	border-top-width: thin;
	border-right-width: thin;
	border-bottom-width: thin;
	border-left-width: thin;
	border-top-style: none;
	border-right-style: none;
	border-bottom-style: none;
	border-left-style: none;
}
#base #secador {
	background-color: #C36;
	float: left;
	height: 300px;
	width: 498px;
	background-image: none;
	background-position: center center;
	background-repeat: no-repeat;
}
#base #aerogenerador {
	background-color: #3F3;
	float: right;
	height: 300px;
	width: 500px;
}
#base #central {
	background-color: #CF9;
	float: left;
	height: 300px;
	width: 500px;
	background-image:
url(file:///D|/RICHARD/Maestria/Proyecto/Pagina%20web/imagenes/wind_generator_turning_lg_nwm.gif);
	background-repeat: no-repeat;
}
#base #libre {
	float: right;
	height: 300px;
```

```
        width: 500px;
        background-color: #00F;
}
#central #base #espacio {
        float: left;
        height: 300px;
        width: 5px;
}
#base #central2 {
        background-color: #0FC;
        float: right;
        height: 300px;
        width: 498px;
        position: static;
        top: 200px;
}
#central #base #titulo {
        background-color: #3CF;
        border-top-style: solid;
        border-right-style: solid;
        border-bottom-style: solid;
        border-left-style: solid;
}
#central #base #espacio_libre {
        height: 5px;
}
#central #base #division {
        float: left;
        height: 470px;
        width: 945px;
        border-top-style: solid;
        border-right-style: solid;
        border-bottom-style: solid;
        border-left-style: solid;
        border-top-width: thin;
        border-right-width: thin;
        border-bottom-width: thin;
        border-left-width: thin;
        position: absolute;
        top: 41px;
}
#central #base #secador2 {
        height: 300px;
        width: 4px;
        float: left;
}
#central #base #espacio2 {
        float: none;
        height: 470px;
        width: 945px;
        position: absolute;
        left: 964px;
        top: 524px;
        border-top-width: thin;
        border-right-width: thin;
        border-bottom-width: thin;
        border-left-width: thin;
        border-top-style: solid;
        border-right-style: solid;
        border-bottom-style: solid;
        border-left-style: solid;
}
```

```
#central #base #central3 {
	height: 470px;
	width: 945px;
	border-top-width: thin;
	border-right-width: thin;
	border-bottom-width: thin;
	border-left-width: thin;
	border-top-style: solid;
	border-right-style: solid;
	border-bottom-style: solid;
	border-left-style: solid;
	position: absolute;
	left: 8px;
	top: 524px;
}
#central #base #aerogenerador3 {
	height: 470px;
	width: 945px;
	border-top-width: thin;
	border-right-width: thin;
	border-bottom-width: thin;
	border-left-width: thin;
	border-top-style: solid;
	border-right-style: solid;
	border-bottom-style: solid;
	border-left-style: solid;
	position: absolute;
	left: 964px;
	top: 41px;
}
#apDiv1 {
	position: absolute;
	left: 14px;
	top: 225px;
	width: 935px;
	height: 286px;
	z-index: 1;
}
#apDiv2 {
	position: absolute;
	left: 358px;
	top: 469px;
	width: 422px;
	height: 35px;
	z-index: 1;
}
#apDiv3 {
	position: absolute;
	left: 407px;
	top: 28px;
	width: 506px;
	height: 38px;
	z-index: 1;
}
#apDiv {
	position: absolute;
	left: 1335px;
	top: -16px;
	width: 497px;
	height: 38px;
	z-index: 1;
}
```

```
#apDiv4 {
	position: absolute;
	left: 11px;
	top: 409px;
	width: 443px;
	height: 38px;
	z-index: 1;
}
#apDiv5 {
	position: absolute;
	left: 16px;
	top: 481px;
	width: 91px;
	height: 156px;
	z-index: 1;
}
#apDiv6 {
	position: absolute;
	left: 789px;
	top: -2px;
	width: 127px;
	height: 104px;
	z-index: 1;
}
#apDiv7 {
	position: absolute;
	left: 215px;
	top: 652px;
	width: 510px;
	height: 38px;
	z-index: 1;
}
#apDiv8 {
	position: absolute;
	left: 164px;
	top: 302px;
	width: 605px;
	height: 38px;
	z-index: 1;
}
#apDiv9 {
	position: absolute;
	left: 219px;
	top: 12px;
	width: 501px;
	height: 81px;
	z-index: 1;
}
#apDiv10 {
	position: absolute;
	left: 613px;
	top: 408px;
	width: 311px;
	height: 39px;
	z-index: 1;
}
#apDiv11 {
	position: absolute;
	left: 294px;
	top: 86px;
	width: 338px;
	height: 38px;
```

```
	z-index: 1;
}
#apDiv12 {
	position: absolute;
	left: 405px;
	top: 43px;
	width: 280px;
	height: 42px;
	z-index: 1;
}
#apDiv13 {
	position: absolute;
	left: 405px;
	top: 81px;
	width: 279px;
	height: 47px;
	z-index: 1;
}
#apDiv14 {
	position: absolute;
	left: 405px;
	top: 121px;
	width: 298px;
	height: 38px;
	z-index: 1;
}
#apDiv15 {
	position: absolute;
	left: 1363px;
	top: 41px;
	width: 235px;
	height: 38px;
	z-index: 1;
}
#apDiv16 {
	position: absolute;
	left: 407px;
	top: 82px;
	width: 234px;
	height: 38px;
	z-index: 1;
}
#apDiv17 {
	position: absolute;
	left: 407px;
	top: 122px;
	width: 250px;
	height: 42px;
	z-index: 1;
}
#apDiv18 {
	position: absolute;
	left: 340px;
	top: 522px;
	width: 267px;
	height: 42px;
	z-index: 1;
}
#apDiv19 {
	position: absolute;
	left: 340px;
	top: 562px;
```

```
        width: 268px;
        height: 42px;
        z-index: 1;
}
#apDiv20 {
        position: absolute;
        left: 340px;
        top: 600px;
        width: 257px;
        height: 38px;
        z-index: 1;
}
#apDiv21 {
        position: absolute;
        left: 12px;
        top: 330px;
        width: 926px;
        height: 134px;
        z-index: 1;
}
#apDiv22 {
        position: absolute;
        left: 7px;
        top: 320px;
        width: 929px;
        height: 144px;
        z-index: 1;
}
#apDiv23 {
        position: absolute;
        left: 669px;
        top: 88px;
        width: 130px;
        height: 30px;
        z-index: 1;
}
#apDiv24 {
        position: absolute;
        left: 661px;
        top: 50px;
        width: 148px;
        height: 63px;
        z-index: 1;
}
#apDiv25 {
        position: absolute;
        left: 1588px;
        top: 9px;
        width: 130px;
        height: 51px;
        z-index: 1;
}
#apDiv26 {
        position: absolute;
        width: 130px;
        height: 12px;
        z-index: 2;
        left: 632px;
        top: 48px;
}
#apDiv27 {
        position: absolute;
```

```
	width: 926px;
	height: 141px;
	z-index: 1;
	left: 10px;
	top: 182px;
}
#apDiv28 {
	position: absolute;
	width: 929px;
	height: 135px;
	z-index: 1;
	left: 7px;
	top: 172px;
}
#apDiv29 {
	position: absolute;
	width: 130px;
	height: 64px;
	z-index: 1;
	left: 632px;
	top: 89px;
}
#apDiv30 {
	position: absolute;
	width: 130px;
	height: 62px;
	z-index: 1;
	left: 670px;
	top: 12px;
}
#apDiv31 {
	position: absolute;
	width: 130px;
	height: 30px;
	z-index: 1;
	left: 560px;
	top: 46px;
}
#apDiv32 {
	position: absolute;
	left: 560px;
	top: 83px;
	width: 130px;
	height: 30px;
	z-index: 1;
}
#apDiv33 {
	position: absolute;
	left: 560px;
	top: 6px;
	width: 130px;
	height: 30px;
	z-index: 1;
}
#apDiv34 {
	position: absolute;
	left: 250px;
	top: 49px;
	width: 130px;
	height: 30px;
	z-index: 1;
}
```

```
#apDiv35 {	position: absolute;
	left: 710px;
	top: 104px;
	width: 130px;
	height: 30px;
	z-index: 1;
	background-color: #CCCCCC;
}
#apDiv36 {	position: absolute;
	width: 130px;
	height: 30px;
	z-index: 2;
	left: 709px;
	top: 143px;
	background-color: #CCCCCC;
}
#apDiv37 {	position: absolute;
	width: 130px;
	height: 30px;
	z-index: 1;
	left: 709px;
	top: 97px;
	background-color: #CCCCCC;
}
#apDiv38 {	position: absolute;
	width: 130px;
	height: 30px;
	z-index: 1;
	left: 642px;
	top: 102px;
	background-color: #CCCCCC;
}
#apDiv39 {	position: absolute;
	width: 130px;
	height: 30px;
	z-index: 1;
	left: 576px;
	top: 100px;
	background-color: #CCCCCC;
}
#apDiv40 {	position: absolute;
	width: 130px;
	height: 30px;
	z-index: 1;
	left: 576px;
	top: 98px;
	background-color: #CCCCCC;
}
#apDiv41 {	position: absolute;
	left: 681px;
	top: 104px;
	width: 130px;
	height: 30px;
	z-index: 1;
	background-color: #CCCCCC;
}
#apDiv42 {	position: absolute;
	left: 12px;
	top: 330px;
	width: 926px;
	height: 134px;
	z-index: 1;
```

```
}
#apDiv42 {          position: absolute;
            left: 12px;
            top: 330px;
            width: 926px;
            height: 134px;
            z-index: 1;
}
</style>
</head>
<body id="central">
<div id="apDiv3">
  <h1>Secador Híbrido Solar - Eólico</h1>
</div>
<div id="base">
  <div id="titulo">   Centro de monitoreo en tiempo real del CIDTER</div>
  <div id="espacio_libre"></div>
  <div id="division"><img src="imagenes/Secadoreolicohibrido.jpg" width="285" height="177" />
    <div id="apDiv">
      <h1>Central Fotovoltaica de 2.5 kW</h1>
    </div>
    <div id="apDiv5">
      <h3><img src="imagenes/unicach.jpg" width="82" height="120" /></h3>
    </div>
    <div id="apDiv7">
      <h1>Centro de Investigación y Desarrollo Tecnológico en Energías Renovables</h1>
    </div>
    <div id="apDiv12">
      <h2>Temperatura del secador:</h2>
    </div>
    <div id="apDiv13">
      <h2>Humedad del secador:
      </h2>
    </div>
    <div id="apDiv14">
      <h2>Temperatura del tanque:</h2>
    </div>
    <div id="apDiv15">
      <h2>Voltaje de entrada:</h2>
    </div>
    <div id="apDiv23" div style=" font-size:24px;">
      <h6> 
        <table width="107" height="24" border="1" align="center" id="ttsecador">
          <tr>
            <td></td>
          </tr>
        </table>
    </div>
    <div id="apDiv24" div style=" font-size:24px;">
      <h6> 
        <table width="107" height="24" border="1" align="center" id="hsecador">
          <tr>
            <td></td>
          </tr>
      </table>
    </div>
    <div id="apDiv25" div style=" font-size:24px;">
      <h6> 
        <table width="107" height="24" border="1" align="center" id="vcentral">
          <tr>
            <td></td>
          </tr>
```

```
    </table>
      <br />
  </div>
  <div id="apDiv30" div style=" font-size:24px;">
    <h6> 
      <table width="107" height="24" border="1" align="center" id="tsecador">
        <tr>
          <td></td>
        </tr>
      </table>
  </div>
  <p> </p>
  <p> </p>
</div>
<div id="espacio2"><img src="imagenes/wind_generator_turning_lg_nwm.gif" width="142" height="166" />
  <div id="apDiv28">
  </div>
  <div id="apDiv22">
  </div>
  <div id="apDiv31" div style=" font-size:24px;">
    <h6> 
      <table width="107" height="24" border="1" align="center" id="caerogenerador">
        <tr>
          <td></td>
        </tr>
      </table>
  </div>
  <div id="apDiv32" div style=" font-size:24px;">
    <h6> 
      <table width="107" height="24" border="1" align="center" id="paerogenerador">
        <tr>
          <td></td>
        </tr>
      </table>
  </div>
  <div id="apDiv33" div style=" font-size:24px;">
    <h6> 
      <table width="107" height="24" border="1" align="center" id="vaerogenerador">
        <tr>
          <td></td>
        </tr>
      </table>
  </div>
</div>
<div id="central3">
  <div id="apDiv6">
    <h3><img src="imagenes/cidter_lineas.jpg" width="153" height="104" /></h3>
  </div>
  <div id="apDiv8">
    <h1>Universidad de Ciencias y Artes de Chiapas</h1>
  </div>
  <div id="apDiv9">
    <h1>Centro de  monitoreo en tiempo real inalámbrico de sistemas energéticos</h1>
  </div>
  <div id="apDiv4">
    <h2>Tuxtla Gutiérrez, Chiapas. México</h2>
  </div>
  <div id="apDiv10">
    <h4>
      <iframe scrolling="no" frameborder="no" clocktype="html5"
style="overflow:hidden;border:0;margin:0;padding:0;width:300px;height:30px;"src="http://www.clocklink.com/ht
```

```
ml5embed.php?clock=018&timezone=Mexico_MexicoCity&color=gray&size=340&Title=&Message=&Target=&
From=2016,1,1,0,0,0&DateFormat=dd / MMM / yy&TimeFormat=hh:mm:ss TT&Color=gray"></iframe>
      </h4>
    </div>
    <div id="apDiv11">
      <h1>renovables del  CIDTER </h1>
    </div>
    <h6> </h6>
  </div>
  <div id="aerogenerador3"><img src="imagenes/panel-solar-600x293.jpg" width="243" height="176" />
    <div id="apDiv2">
      <h1>Aerogenerador de 900 W</h1>
    </div>
    <div id="apDiv16">
      <h2>Corriente de entrada:</h2>
    </div>
    <div id="apDiv17">
      <h2>Potencia consumida:</h2>
    </div>
    <div id="apDiv18">
      <h2>Voltaje de entrada:</h2>
    </div>
    <div id="apDiv19">
      <h2>Corriente de entrada:</h2>
    </div>
    <div id="apDiv20">
      <h2>Potencia consumida:</h2>
    </div>
    <div id="apDiv21">

    </div>
    <div id="apDiv26" div style=" font-size:24px;">
      <h6> 
        <table width="107" height="24" border="1" align="center" id="ccentral">
          <tr>
            <td></td>
          </tr>
        </table>
    </div>
    <div id="apDiv29" div style=" font-size:24px;">
      <h6> 
        <table width="107" height="24" border="1" align="center" id="pcentral">
          <tr>
            <td></td>
          </tr>
        </table>
    </div>
    <div id="apDiv27"></div>
    <p> </p>
  </div>
  <p> </p>
</div>
        <script type="text/javascript" src="jquery.js"></script>
        <script type="text/javascript" src="Chart.js"></script>
<script type="text/javascript" src="Chart.min.js"></script>
<script type="text/javascript" src="canvasjs.min.js"></script>
        <script type="text/javascript" src="jquery-2.2.3.min.js"></script>
<div id="apDiv1">
</div>
<script type="text/javascript">
            window.onload = function () {
                var dataLength = 0;
```

```
                                    var dataLength1 = 0;
                                    var dataLength2 = 0;
        var data = [];
                                    var data1 = [];
                                    var data2 = [];
                                    var data3 = [];
                                    var data4 = [];
                                    var data5 = [];
                                    var data6 = [];
                                    var data7 = [];
                                    var data8 = [];
        var updateInterval = 500;
        updateChart();

function updateChart() {

            $.getJSON("central.php", function (result) {
                if (dataLength !== result.length) {
                                                        for (var i = 1; i >= 0; i--) {
                                                            //var fecha = new Date();
                                                            data.push({
                                                                label: result[i].horacentral,
                        y: parseInt(result[i].voltajec),
                                                            });
                                                            data1.push({
                                                                label: result[i].horacentral,
                        y: parseInt(result[i].corrientec),
                                                            });
                                                            data2.push({
                                                                label: result[i].horacentral,
                        y: parseInt(result[i].potenciac),

                                                            });
                    }
                    dataLength = result.length;
                    chart.render();
                                                        chart1.render();
                }
            });
                                        $.getJSON("aerogenerador.php", function (result) {
                if (dataLength1 !== result.length) {
                                                        for (var j = 1; j >= 0; j--) {
                                                            data3.push({
                                                                label: result[j].horaaero,
                        y: parseInt(result[j].voltajea),
                                                            });
                                                            data4.push({
                                                                label: result[j].horaaero,
                        y: parseInt(result[j].corrientea),
                                                            });
                                                            data5.push({
                                                                label: result[j].horaaero,
                        y: parseInt(result[j].potenciaa),
                                                            });
                    }
                    dataLength1 = result.length;
                    chart2.render();
                                                        chart3.render();
                }
            });
                                    $.getJSON("secador.php", function (result) {
                if (dataLength2 !== result.length) {
```

```
                                                                    for (var k = 1; k >= 0; k--) {
                                                                        //var fecha = new Date();
                                                                        data6.push({
                                                                            label: result[k].horasecador,
                        y: parseInt(result[k].tsecador),
                                                                    });
                                                                    data7.push({
                                                                            label: result[k].horasecador,
                        y: parseInt(result[k].hsecador),
                                                                    });
                                                                    data8.push({
                                                                            label: result[k].horasecador,
                        y: parseInt(result[k].ttsecador),
                                                                    });
                    }
                    dataLength2 = result.length;
                    chart4.render();
                }
            });
        }

        var chart = new CanvasJS.Chart("apDiv27", {
                                        width: 890,
                                        height: 145,
            axisX: {
                        labelFontSize: 14
            },
            axisY: {
                title: "Voltaje (V)",
                        titleFontSize: 14,
                        labelFontSize: 14,
                        lineColor: "#369EAD"
            },
                    axisY2:{
                title: "Corriente (A)",
                        titleFontSize: 14,
                labelFontSize: 14,
                        maximum: 100,
                        lineColor: "#C24642"
    },
            data: [{type: "spline", dataPoints: data, xValueType: "String", },
                                                {type:  "spline",  axisYType:  "secondary",  dataPoints:  data1,
xValueType: "String", }],
        });

                                    var chart1 = new CanvasJS.Chart("apDiv21", {
                                        width: 850,
                                        height: 140,
            axisX: {
                        labelFontSize: 14
            },
            axisY: {
                title: "Potencia (W)",
                        titleFontSize: 14,
                        labelFontSize: 14
            },
            data: [{type: "spline", dataPoints: data2, xValueType: "String", }],
        });
                                    var chart2 = new CanvasJS.Chart("apDiv28", {

                                        width: 890,
                                        height: 155,
```

```
            axisX: {
                        labelFontSize: 14
            },
            axisY: {
               title: "Voltaje (V)",
                        titleFontSize: 14,
                        labelFontSize: 14,
                        lineColor: "#369EAD"
            },
                                    axisY2:{
      title: "Corriente (A)",
                titleFontSize: 14,
          labelFontSize: 14,
                maximum: 20,
                lineColor: "#C24642"
    },
            data: [{type: "spline", dataPoints: data3, xValueType: "String", },
                                        {type: "spline", axisYType: "secondary", dataPoints: data4,
xValueType: "String", }],
          });
                                var chart3 = new CanvasJS.Chart("apDiv22", {
                                    width: 850,
                                    height: 150,
            axisX: {
                        labelFontSize: 14
            },
            axisY: {
               title: "Potencia (W)",
                        titleFontSize: 14,
                        labelFontSize: 14
            },
            data: [{type: "spline", dataPoints: data5, xValueType: "String", }],
          });
                                var chart4 = new CanvasJS.Chart("apDiv1", {
                                    width: 890,
                                    height: 280,
            axisX: {
                        labelFontSize: 14
            },
            axisY: {
               title: "Temperatura (°C)",
                     labelFontSize: 14,
                        lineColor: "#369EAD"
            },
                                    axisY2:{
      title:   "Humedad (%)",
                    maximum: 100,
                    lineColor: "#C24642"
                                    },

            data: [{type: "spline", dataPoints: data6, xValueType: "String", },
                                        {type: "spline", axisYType: "secondary", dataPoints: data7,
xValueType: "String", },
                                                {type: "spline", dataPoints: data8, xValueType: "String",
}],
          });
          setInterval(function () {updateChart()}, updateInterval);
        }
      </script>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script type="text/javascript">
   function getTimeAJAX() {
```

```
        var time = $.ajax({
            url: 'vcentral.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("vcentral").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">

    function getTimeAJAX() {
        var time = $.ajax({
            url: 'ccentral.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("ccentral").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'pcentral.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("pcentral").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'tsecador.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("tsecador").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'hsecador.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("hsecador").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'ttsecador.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("ttsecador").innerHTML = time;
```

```
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'vaerogenerador.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("vaerogenerador").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'caerogenerador.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("caerogenerador").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
<script type="text/javascript">
    function getTimeAJAX() {
        var time = $.ajax({
            url: 'paerogenerador.php', //indicamos la ruta donde se genera la hora
                dataType: 'text',//indicamos que es de tipo texto plano
                async: false     //ponemos el parámetro asyn a falso
        }).responseText;
        document.getElementById("paerogenerador").innerHTML = time;
    }
    setInterval(getTimeAJAX,1000);
</script>
</body>
</html>
<?php
?>
```