UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA ELÉCTRICA

Construcción de medidores trifásicos Inalámbricos de consumo de energía

PRESENTADO POR:

ALVARO ANTONIO CALDERÓN MAGAÑA

CARLOS ENRIQUE MOLINA FLAMENCO

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, SEPTIEMBRE DE 2012

UNIVERSIDAD DE EL SALVADOR

RECTOR :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIA GENERAL :

DRA. ANA LETICIA ZAVALETA DE AMAYA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR :

ING. JOSÉ WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

Construcción de medidores trifásicos inalámbricos de consumo de energía

Presentado por :

ALVARO ANTONIO CALDERON MAGAÑA

CARLOS ENRIQUE MOLINA FLAMENCO

Trabajo de Graduación Aprobado por :

Docente Director :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

San Salvador, Septiembre de 2012

Trabajo de Graduación Aprobado por:

Docente Director :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

ACRÓNIMOS.

-EIE: Escuela de Ingeniería Eléctrica.

-UES: Universidad de El Salvador.

-AP: Punto de acceso (del inglés access point).

-MCU: Unidad microcontroladora (del inglés microcontroller unit).

-PIC: Controlador de interfaz periférico (del inglés peripheral interface controller).

-SPUD: Panel de control simple unificado (del inglés simple unified dashboard).

-GUI: Interfaz gráfica de usuario (del inglés graphical user interface) .

-PCB: Tarjeta de Circuito Impreso (del inglés printed circuit board).

- ADC: Convertidor analógico digital (del inglés analog to digital converter).

- BCD: Decimal codificado en binario (del inglés binary-coded decimal).

- UART: Transmisor-receptor asíncrono universal (del inglés universal asynchronous receiver-transmitter).

- I2C: Circuito Inter-Integrados (del inglés inter-integrated circuit).

- RTC:Reloj de Tiempo Real (del inglés Real Time Clock).

- WIFI: Fidelidad inalámbrica (del inglés wireless fidelity).

- NTP: Protocolo de tiempo de red (del inglés network time protocol).

- RRDtool: Herramienta de base de datos de round robin (del inglés round robin database Tool).

- B.A.T.M.A.N.: Better Approach To Mobile Adhoc Networking.

- B.A.T.M.A.N.D.: Better Approach To Mobile Adhoc Networking Daemon.

INDICE

LISTA DE TABLAS

LISTA DE FIGURAS

Capítulo 1

Desarrollo y evolución del medidor de energía de bajo consumo.

1.1. Introducción

Para hacer uso eficiente de la energía eléctrica es necesario conocer el perfil de carga de una instalación. Es en este campo en el que esta investigación trata de hacer su aporte. El diseño de un medidor de bajo costo que tiene como finalidad dar información sobre la tendencia en el consumo de energía eléctrica. La filosofía del diseño propuesto es reducir los costos de manufacturación, esto se logra con el uso de herramientas de software libre y circuitería sencilla en el hardware. Sacrificando la exactitud de las mediciones ya solamente es censada la corriente consumida. Se fija el parámetro de voltaje a 120Vrms (linea a neutro).

Además, el diseño de estos medidores contempla su funcionamiento dentro de un entorno de red inalámbrica tipo malla usando tecnología WIFI. La implementación de un servidor web que permite almacenar las mediciones de energía y potencia eléctricas censadas, además de desplegar de forma gráfica dichas mediciones

1.2 Intereses y motivaciones de la investigación

Actualmente en todo el mundo se vive una crisis energética debido a los altos costos de los combustibles fósiles. Los gobiernos han lanzado grandes campañas que llaman a la población ha hacer uso razonado de la energía. Con esta idea en mente y con el deseo de aportar una solución a este problema social, este proyecto trata de mejorar una idea que ha venido siendo desarrollada en diversos trabajos de graduación anteriores dentro de la EIE de la UES.

1.3 Objetivos

Objetivo general
Manufacturación de dispositivos electrónicos de bajo costo.

Objetivo específico
Diseño de medidores trifásicos de consumo de energía.
Construcción de prototipos de medidores de consumo de energía.
Configuración de medidores en topología de red tipo malla.
Monitorización remota de parámetros de enlace de radio.
Monitorización remota de consumo de energía eléctrica.

1.4 Organización

El presente documento se encuentra organizado en 4 capítulos cada uno de los cuales se subdivide en distintas secciones (algunas con apartados). A modo de ejemplo el primer capitulo titulado Desarrollo y evolución del medidor de energía de bajo consumo tiene cinco secciones 1.1, 1.2, 1.3, 1.4 y 1.5

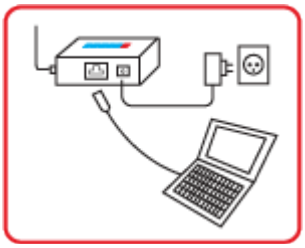
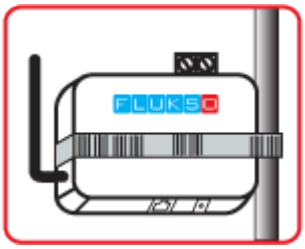
Cada sección puede incluir elementos adicionales como son figuras ilustrativas, tablas, segmentos de código o ecuaciones. Estos elementos se encuentran diferenciados por la sección del documento en que se encuentran seguidos de un número que ha sido asignado en el orden en que estos aparecen. Por ejemplo, la tabla 1.5.1 es la primer tabla que aparece en la sección 1.5 y la Figura 1.5.1 es la primera ilustración que se muestra en la misma sección. Para hacer referencia a elementos dentro de los Anexos se hará uso de la letra correspondiente al Anexo y un número para identificar el elemento. Por ejemplo Figura E.1 hace referencia al primer segmento de código dentro del Anexo E.

1.5 Antecedentes

La idea de un medidor de bajo costo fue desarrollada por Flukso [14]. Esta empresa con sede en Bélgica, propone con su medidor garantizar la reducción del consumo energético residencial por medio de la monitorización de la potencia eléctrica consumida. Se busca que el usuario haga uso mas eficiente de la energía consumida al ser consciente de su consumo. La visualización del consumo se hace a través de una gráfica de la potencia consumida en función del tiempo. El hardware esta basado en un circuito sensor empotrado en un router WAP-2102 de Abocon.

El medidor de flukso necesita acceso a internet para poder enviar los datos medidos hacia el sitio web de flukso, donde las mediciones se almacenan en una base de datos. Cada usuario puede ver la tendencia de consumo. El acceso a internet puede ser provisto por un AP que hace el papel de gateway para darle salida a la web, este puede ser perfectamente un router inalámbrico de cualquier compañía que provea acceso a internet.

Flukso facilita el trabajo de instalación de sus dispositivos y simplifica los pasos para hacerlo.

	Paso 1. Configuración WIFI: Conectar el medidor a un toma corriente, luego conectar con cable de red a un computador. Desde un navegador web acceder a la dirección IP estática del medidor 192.168.255.1 y configurar el nombre y clave de la red inalámbrica con acceso a internet a la que el medidor se conectará.
	Paso 2. Asegurar el medidor: Desconectar todos los cables del medidor y asegurar el medidor en un lugar fijo cerca de la caja térmica de la instalación que se pretende monitorizar.

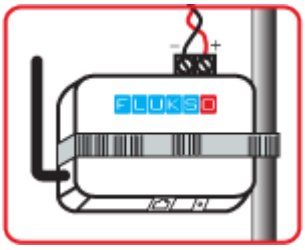
	Paso 3. pinzas de corriente: Por razones de seguridad desconectar el cortacircuitos principal en la caja térmica luego colocar la pinza alrededor de una fase y cerrar la pinza
	Paso 4. cable de las pinzas: conectar los cables de las pinzas de corriente al medidor respetando la polaridad de los mismos
	Paso 5. encender el sistema: conectar el cable de alimentación del medidor nuevamente y acceder a la URL https://www.flukso.net/user/register para dar de alta el medidor y poder ver los datos medidos.

Tabla 1.5.1 *Instalación medidor Flukso [14].*

La Tabla 1.5.1 muestra los 5 pasos para la instalación de un medidor flukso. La instalación es muy sencilla y no requiere de conocimientos técnicos avanzados. Es importante mencionar que estos medidores estan pensados para funcionar en una red inalámbrica WIFI, requiriendo de un punto de acceso a internet.

Conociendo que el medidor de flukso usa software libre y sus desarrolladores han compartido toda la información para hacer funcionar el medidor, nace la idea de desarrollar una copia de este en la escuela de ingeniería eléctrica. Los cambios principales fueron: uso de router dir-300 y uso de microcontrolador PIC-16F876. Para lograr el primer prototipo de un medidor monofásico de energía electrica se hizo en 3 etapas que duraron alrededor de 2 años, donde cada una de estas son: proyecto de ingeniería 1, proyecto de ingeniería 2 y proyecto de graduación [15]. Con este proyecto se logró reducir el tamaño del circuito impreso del medidor, identificar tiendas en internet para la compra e importación de los elementos electrónicos necesitados, conocer fabricantes de PCB's.

El medidor desarrollado en la UES era para uso residencial en instalaciones monofásicas, el objetivo de este es presentar una tendencia del consumo, por lo que se censaba la corriente y el parámetro de voltaje se dejaba con el valor constante de 120V. La corriente era medida a través de un transductor de corriente, se usaba una pinza de corriente de 50 amperios con una resolución de 1Amp-AC/ 100mV-DC (mismo sensor usado por flukso), este valor de voltaje DC convertido se lleva a la entrada del ADC del PIC para ser digitalizado, luego el valor de corriente se multiplica por 120 voltios y se obtiene la potencia. Luego con procesos de retardo se calcula la energía consumida. Estos datos son enviados a través del puerto UART del PIC hacia el puerto UART del router DIR-300. Dentro del router los datos son capturados por el programa flukso.lua que después de hacer unos cálculos, envía los datos hacia el

servidor a través de XML/RPC. Además se incluyó un circuito reloj de tiempo real RTC1307, esto sirve para conocer el tiempo en que las mediciones son enviadas.

El formato de los datos enviados por el puerto UART se muestra en la Tabla 1.5.2

COMANDO	ID	MEDICIÓN	FIN	PARÁMETRO
pwr	2d4f76edc4dcf577fa0a49b8335l3bfe	:0000000120	0A	potencia
pls	2d4f76edc4dcf577fa0a49b8335l3bfe	:0000000300	0A	energía
tim	2d4f76edc4dcf577fa0a49b8335l3bfe	:1110011057	0A	tiempo

Tabla 1.5.2 *Formato de datos enviados por el PIC al puerto UART*

El router DIR-300 del medidor desarrollado en la UES [15] al igual que el router WAP-2102 del medidor de flukso, estaba configurado en modo-STA, por lo que éste necesitaba de otro router inalámbrico o un AP para enviar los datos al servidor de forma satisfactoria.

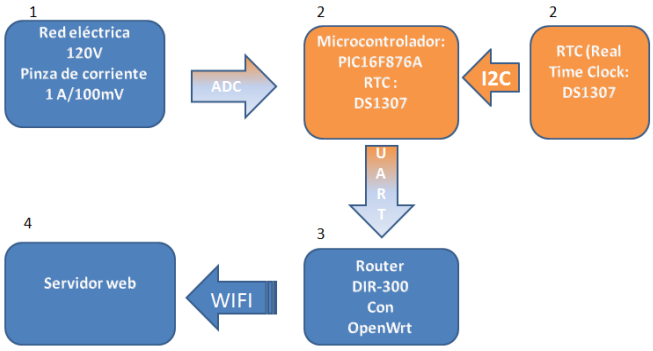


Figura 1.5.1 *Bloques Medidor de consumo eléctrico de bajo costo[15].*

Como es posible apreciar en la Figura 1.5.1 el funcionamiento de este medidor es muy parecido al del medidor de flukso. Primero la corriente es censada por las pinzas y la señal es convertida a voltaje DC el cual es digitalizado por el MCU este toma la fecha y hora del RTC usando el puerto i2c y envía los datos calculados de potencia y energía al puerto UART del router Dir-300 quien se encarga de enviar los datos promediados cada 5 minutos por medio de WIFI y protocolo XML-RPC al servidor web para su almacenamiento y posterior visualización. Dicho servidor web utiliza apache 2 en una plataforma Linux. Contiene las bases de datos de los medidores (RRDTool) donde se almacenan los datos enviados por los medidores, además de la base de datos de usuarios asociados a los medidores (MySQL). Para la visualización de las mediciones se usa la herramienta RRDTOOL. Contiene las CGI necesarias para realizar las funciones de: ingresar usuario, crear un nuevo usuario, generar y visualizar las gráficas de las mediciones, estas CGI programadas en PHP y comandos del SHELL de Linux. Contiene los procedimientos remotos, cuyo acceso se realiza por medio de XML- RPC (implementado en PERL).

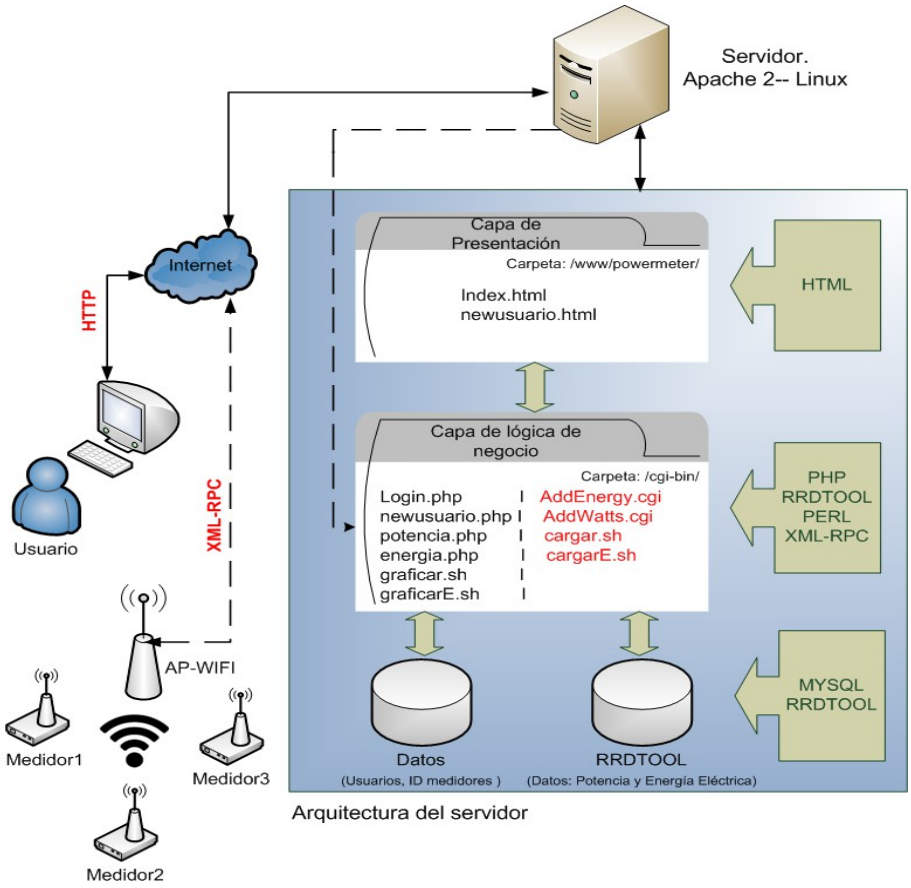


Figura 1.5.2 *Arquitectura del Servidor Web [15]*

La Figura 1.5.2 muestra el modelo de 3 capas del servidor implementado para el medidor desarrollado en la UES[15], la capa de presentación, capa de negocios y capa de datos. La presentación se hace por medio de HTML la capa de negocios usa una combinación de PHP, PERL, RRDTool y XML-RPC, dentro de la capa de datos se tienen dos bases distintas una que sirve para guardar los datos de usuario (nombre del usuario, ID del medidor, correo electrónico del usuario, etc) esta ha sido implementada en MySQL y la base de datos que sirve para almacenar los datos medidos de energía y potencia implementada en RRDTool.

1.6 Descripción general del medidor trifásico de consumo de energía.

Con la finalidad de darle continuidad al medidor monofásico descrito en la sección 1.5, se plantea hacer las siguientes mejoras al diseño del medidor: Capacidad trifásica, capacidad de funcionar en una red de medidores inalámbrica configuración malla, implementación de un servidor web con capacidad de monitorizar los enlaces dentro de la red a la vez de almacenar y desplegar los datos medidos.

En este trabajo de graduación se realizó la manufactura de 5 medidores trifásicos y su configuración en una red tipo malla. Para mantener los costos de manufacturación bajos se siguió con la metodología del primer prototipo[15] de sensar solamente la corriente y dejar el parámetro de voltaje fijo en un valor de 120 Vrms. El corazón del circuito sensor de corriente es un PIC16F876A quien se encarga de convertir los datos analógicos generados por la pinza de corriente a información digital. A partir de esos datos se

calculan las variables eléctricas de potencia y energía. Además se siguió usando un circuito RTC quien se comunica con el PIC por medio del puerto i2c. La información de potencia, energía consumida y hora son enviados por el puerto UART al router Dir-300 quien ha sido configurado para promediar y enviar estos datos usando el protocolo XML-RPC al servidor web para la visualización y la persistencia de estos. Para poder empotrar el circuito sensor dentro del router Dir-300 se hizo uso de dispositivos de montaje superficial. Además se modificó el esquema original del PCB del primer prototipo monofásico [15] para poder dotar al nuevo diseño de la capacidad trifásica (el archivo del nuevo esquema esta en el CD adjunto). Entre los experimentos realizados se encuentra la sustitución de la fuente de voltaje de alimentación del circuito sensor de 5VDC provenientes de la alimentación del Dir-300, por 3.3VDC tomados del puerto UART, dando resultados satisfactorios en el funcionamiento del equipo. También se experimento suprimiendo el circuito RTC por el servicio NTP, siendo este instalado en el servidor web. Este trabajo de graduación pretende hacer otro aporte con la experiencia en la manufactura de este tipo de electrónica y la documentación para que cualquier empresa pueda comercializar esta idea dentro del mercado nacional.

Para monitorizar la red red inalámbrica y poder visualizar los datos medidos por los medidores se hizo uso de una herramienta de software libre llamada SPUD. Esta es una aplicación web que tiene una GUI donde se puede visualizar los miembros de una red tipo malla y los enlaces activos dentro de ella. Además posee la base de datos con la información de los nodos de la red. En este trabajo de graduación se dispuso la modificación de este software para poder trabajar con los medidores construidos, para lo cual se dotó al sistema de la capacidad de generar bases de datos en RRDTool para las mediciones y el despliegue gráfico de estas dentro de la misma GUI que sirve para el monitoreo de la red.

Capítulo 2

Diseño del circuito medidor trifásico y código del microcontrolador.

2.1 Diseño del circuito medidor trifásico.

Para la manufacturación de los circuitos medidores trifásicos se ha utilizado la menor cantidad de hardware posible. El sensor medidor hace uso de transductores de corriente y evita el uso de transductores de voltaje, se usa un valor de voltaje de 120V en las instrucciones del microcontrolador. El valor de corriente es capturado a través de tres pines ADC del microcontrolador. Debido a que el valor de voltaje es una constante y no será medida para cada carga específica, el medidor incluye un error en las mediciones, error que para nuestros propósitos se considera despreciable debido a que interesa tener una tendencia del consumo energético y no un valor preciso.

Las etapas principales del circuito trifásico son: **1)** transductor de corriente; **2)** microcontrolador; **3)** reloj de tiempo real; **4)** salida de los datos a través del puerto UART.

En el diagrama de bloques representado en la Figura 2.1.1 se puede apreciar la manera de acoplar cada una de las etapas mencionadas.

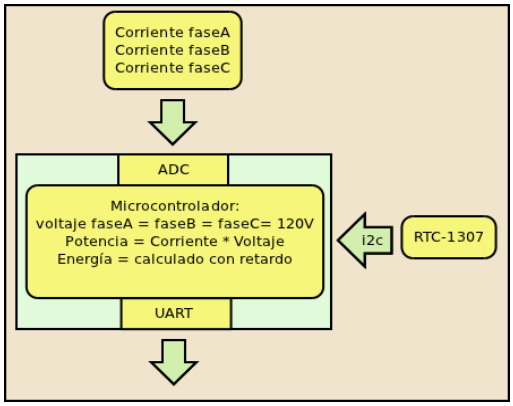


Figura 2.1.1. *Diagrama a bloques del sensor trifásico.*

En el bloque superior de la figura 2.1.1 se representan los transductores de corriente, estas son pinzas que convierten la magnitud de la corriente AC que circula a través del conductor que envuelve la pinza. Esta corriente es convertida a la salida de la pinza en un voltaje DC. La resolución de las pinzas es: 1AAC/100mVDC. Las pinzas usadas soportan un máximo de 50AAC. Por tanto a la salida entregarán un máximo de 5VDC por lo que es posible conectar los terminales de la pinza a cada pin ADC del microcontrolador. El acople se hace utilizando un condensador y una resistencia para limitar la corriente de entrada al pin ADC del microcontrolador.

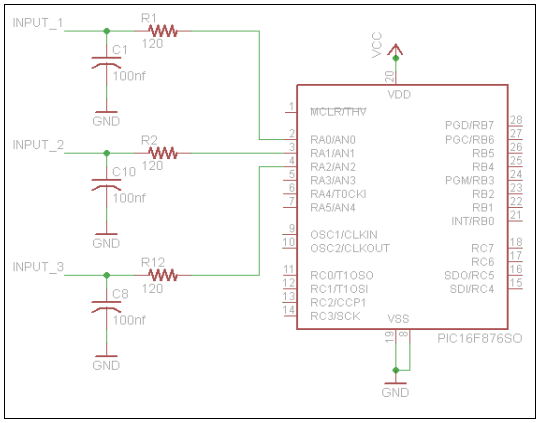


Figura 2.1.2 *Conexión eléctrica entradas ADC.*

La Figura 2.1.2 muestra la interfaz necesaria para conectar cada una de las pinzas de corriente a las entradas ADC del microcontrolador.

En el diagrama a bloques de la Figura 2.1.1 se incluye el circuito integrado RTC-DS1307. Éste es un dispositivo reloj/calendario de bajo consumo de energía completo con código binario decimal (BCD). Los datos son transferidos a través de comunicación serial de 2 hilos usando el protocolo i2c. El reloj/calendario provee información de: segundos, minutos, horas, dia, fecha, mes y año. El final de fecha de mes se ajusta automáticamente durante meses menores de 31 días incluyendo correcciones para el año bisiesto. Este RTC está programado para funcionar hasta el año 2100. El reloj funciona en formato de 24 ó 12 horas con indicador AM/PM. El DS1307 tiene incorporado un circuito de sensor de tensión que detecta fallas de energía y cambia automáticamente al suministro de batería de respaldo, se le puede configurar una salida de onda cuadrada programable, para detalles prácticos del RTC se ha consultado la bibliografía [3].

El tercer bloque de la Figura 2.1.1, representa el microcontrolador. Se ha utilizado el microcontrolador fabricado por microchip PIC-16F876A. De la hoja de datos [4] se obtienen las siguientes características: **1)** frecuencia de reloj de hasta 20MHz; **2)** rango de voltaje de operación desde 2.0V hasta 5.5V; **3)** convertidor analógico digital ADC de 10 bits multicanal; **4)** tres temporizadores; **5)** comunicación por interfaz USART Universal Synchronous Asynchronous Receiver Transmitter; **6)** puerto serie síncrono (SSP) con SPI e I2C. Estas características son suficientes para el uso que se necesita que el microcontrolador realice dentro del circuito sensor de energía trifásica.

El microcontrolador es el encargado de procesar toda la información para calcular la potencia y energía de cada fase. Así como de obtener la hora y fecha del RTC. El PIC envía los datos a través del puerto UART hacia el router DLink DIR-300.

El puerto UART del microcontrolador es acoplado con el puerto UART del router Dlink DIR-300. Para poder enviar los datos necesarios estos están comunicados a una tasa de transferencia de 4800 baudios.

Para el acople mecánico y conección eléctrica de los elementros electrónicos del sensor de energía, se manufacturaron las tabletas de circuito impreso PCB en la empresa FUTURLEC ubicada en Hong Kong. Las características de las PCB´s son las siguientes: **1)** ancho: 6.5cm; **2)** altura: 4.5cm; **3)** pistas impresas a doble cara; **4)** pistas esmaltadas; **5)** superficies de soldaduras estañadas; **6)** pistas unidas con remaches. En la Figura 2.1.3 se muestra ambas caras de la tableta impresa (PCB).

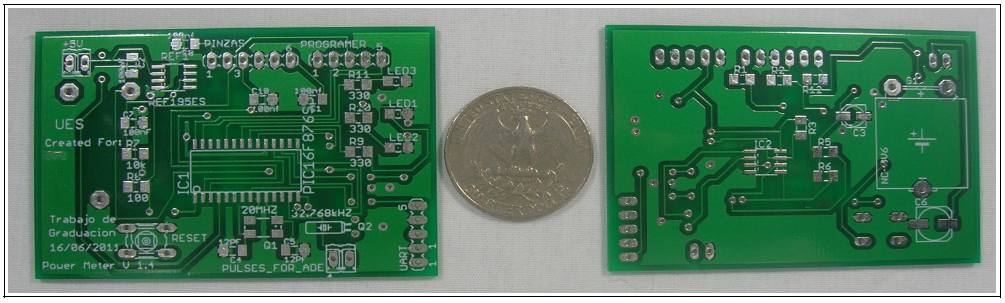


Figura 2.1.3. *Circuito impreso del medidor trifásico.*

El circuito impreso de la Figura 2.1.3 ha sido diseñado para usar elementos electrónicos de soldadura superficial. Todos los contactos de soldadura son para circuitos integrados, resistencias, condensadores y cristales. Se dejaron los terminales no superficiales para soldar los pines de conexión externa y los pines de las baterías. Debe tomarse en cuenta que una de las características de estos circuitos impresos, es que las pistas están diseñadas para ser soldadas a una temperatura de entre 200 a 350 grados centígrados.

La soldadura de los elementos electrónicos superficiales se realizaron en una estación de soldadura, con la cual cada dispositivo electrónico es soldado a temperatura óptima, la Tabla 2.1.1 detalla las temperaturas usadas:

Elemento	Temperatura
Diodo led	250°C
Resistencias	300°C
Capacitores	350°C
Circuitos integrados	300°C
Conectores	350°C

Tabla 2.1.1. *Temperaturas para soldadura.*

En el proceso de soldadura se utilizó: **1)** alambre para desoldar; **2)** estaño 60/40; **3)** kester; **4)** limpiador de contactos y thinner; **5)** cepillo dental cerdas suaves; **6)** desolda pull.

Uno de los elementos mas delicados en el proceso de soldadura son los diodos LED. Para su soldadura se debe usar una temperatura baja y ser estañados manteniendo el calor del cautín el menor tiempo posible, esto para evitar que el contacto metálico que une el cátodo con el ánodo se destruya, en la Figura 2.1.4 se aprecia una foto de un diodo led visto a través de un microscopio.

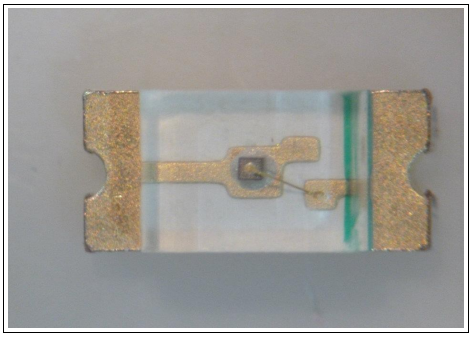


Figura 2.1.4. *Diodo led visto con un microscopio.*

En la Figura 2.1.4 se puede apreciar la línea de color verde que está a la derecha de la imagen, es la marca que representa el cátodo del diodo led, esto es la guía para no equivocarnos de posición en el momento de realizar la soldadura.

La Figura 2.1.5 muestra el circuito terminado. Se puede visualizar que todos los elementos son de soldadura superficial. El conector macho de la placa inferior tiene la función de acoplarse con el puerto UART del router Dlink DIR-300. Los tres conectores de la placa superior son usados para: conector de dos pines: alimentación del circuito a 5 voltios DC. Conector de seis pines: conexión de las pinzas de corriente. Conector de cinco pines: conexión para el programador del microcontrolador. Los remaches que unen las pistas de la cara superior e inferior han sido selladas con estaño para evitar la pronta oxidación de estos.

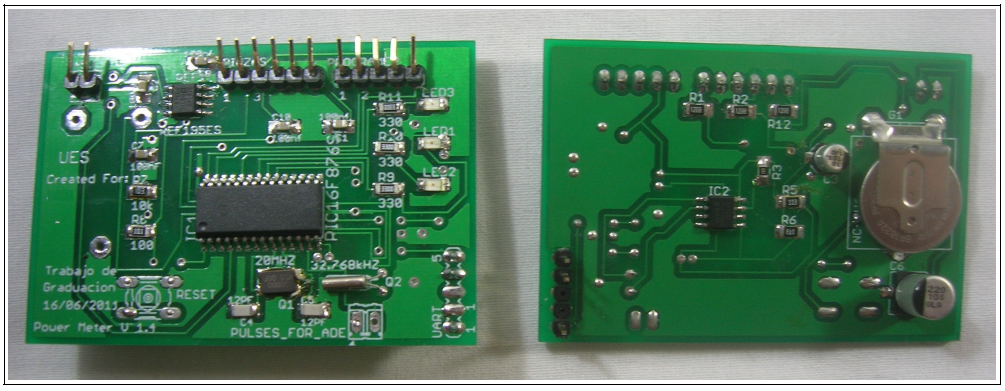


Figura 2.1.5. *Cara superior e inferior del sensor de energía.*

La conexión física entre las pinzas de corriente y el sensor, se hace a través de espigas monaurales, con estas se facilita la conexión de las pinzas eliminando la necesidad de atornillar los cables de las pinzas.

Así como también es posible conectar a 0V cada una de las entradas del ADC para que no registre mediciones cuando esté ausente una pinza de corriente. Cada jack monaural posee tres terminales, uno para conectar 0V o común, otro para conectar la entrada de señal y un tercero que tiene movimiento, el cual es liberado cuando la espiga macho se conecta al jack hembra. Cuando la espiga está desconectada el tercer terminal hace contacto con el terminal de entrada de señal permitiendo llevar a 0V cada una de las entradas del ADC; con este sistema se logra llevar 0V a las entradas que no tengan una pinza de corriente conectada y el valor de voltaje DC de la pinza cuando está conectada al circuito medidor. Esto se ilustra en la Figura 2.1.6.

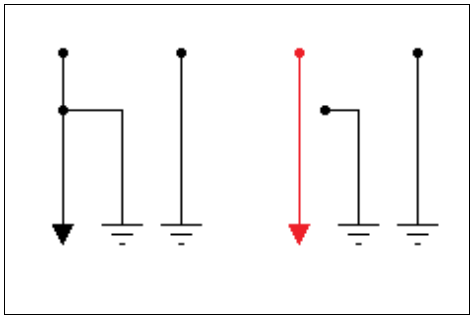


Figura 2.1.6. *a) Espiga macho no conectada. b) Espiga macho conectada.*

La manera de acoplar las pinzas en el medidor se muestra en la Figura 2.1.7. Cada una de las pinzas tiene conexión eléctrica hasta las entradas de cada canal ADC. Cuando una pinza no está conectada el convertidor ADC detecta 0V en la entrada y por tanto el valor de potencia calculado es de 0Watts. De igual manera la energía acumulada por la fase es de 0Wh. Con esta capacidad el medidor trifásico puede usarse en sitemas monofásicos, bifásicos o trifásicos.

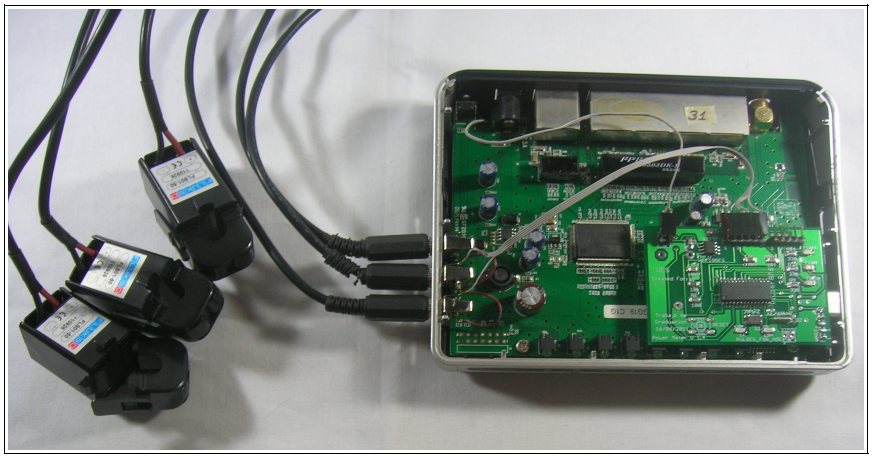


Figura 2.1.7. *Acople de las pinzas en el medidor.*

2.2. Microcontrolador.

Se ha utilizado el microcontrolador PIC-16F876A fabricado por microchip. Éste es el encargado de convertir las señales de los transductores de corriente en datos binarios, hacer el respectivo procedimiento para calcular la potencia por fase y la energía acumulada, obtener la hora del RTC y enviar los datos a través del puerto UART. El PIC utiliza un cristal de 20MHz, con esta frecuencia de trabajo se fija el tiempo en que el PIC procesa cada una de las instrucciones. El programa del microcontrolador se ha desarrollado en lenguaje-C.

La comunicación UART se configura para comunicarse a una tasa de transferencia de 4800 baudios en correspondencia con la tasa de transferencia utilizada y configurada en el router DIR-300. Se usa el puerto de comunicación serial I2C para comunicar el microcontrolador con el RTC-DS1307, el PIC se encuentra configurado en modo maestro y el RTC está en modo esclavo.

Se utiliza el DS1307 para generar una señal de 1Hz que se usa para generar la interrupción externa del PIC. Cada vez que esta interrupción se activa, en las rutinas del microcontrolador se leen los 3 canales ADC del PIC, que representa la corriente que se está midiendo. Luego se procesa para poder obtener la potencia que es enviada por el puerto UART. Se debe tener en cuenta que el valor de voltaje siempre será una constante, para nuestro caso vale 120 voltios. Cada 300 segundos se envía la energía acumulada y cada 600 segundos se lee la hora y fecha del DS1307 y se envía por el puerto UART.

Al iniciar el programa, el PIC hace un retardo de 90 segundos, tiempo calculado en que el router dir-300 inicia todos sus procesos y está listo para poder comunicarse con el sensor de energía trifásico y enviar los datos por la red inalámbrica.

2.2.1. Convertidor Analógico Digital.

El microcontrolador usa un módulo ADC de 10 bits de resolución. El valor del bit menos significativo es función del voltaje de referencia. En este proyecto se usa 0V y 5V como referencia negativa y positiva respectivamente, esto se representa con la ecuación 2.2.1:

$$1\text{LSB} = V_{REF-} + \left(\frac{V_{REF+} - V_{REF-}}{1024}\right) \qquad \text{(ecuación 2.2.1)}$$

Donde:

$V_{REF+} = 5V$

$V_{REF-} = 0V$

Resolviendo se obtiene la resolución en la ecuación 2.2.2:

$$1\text{LSB} = \frac{5}{1024} = 4.8\text{mV} \qquad \text{(ecuación 2.2.2)}$$

La pinza de corriente o transductor de corriente, tiene una resolución de 1A-AC/100mv-DC, si por la pinza circula 1A-AC significa que en el PIN ADC del pic se tendrán 100mV-DC, el valor digitalizado por el PIC se calcula usando la ecuación 2.2.3:

$$valorADC = \frac{100\text{mV}}{4.88\text{mV}} = 20.49 \qquad \text{(ecuación 2.2.3)}$$

Para obtener el valor de corriente se usa la ecuación 2.2.4:

$$Corriente = valorADC * \left(\frac{5}{1024}\right) = 0.10004883\text{Amp} \qquad \text{(ecuación 2.2.4)}$$

Este valor tiene que ser representado como 1Amp, para lograrlo solo falta multiplicar el dato por 10 unidades:

$$0.10004883 * 10 = 1.00048\text{Amp} \qquad \text{(ecuación 2.2.5)}$$

Podemos apreciar que hemos incorporado un poco de error en nuestra lectura y el dato de corriente es mayor que la corriente real, esto puede considerarse aceptable para nuestros propósitos.

2.2.2. Programación del DS1307.

Antes de programar el PIC, se debe ajustar el RTC con un programa dedicado para configurar la hora y fecha de este, hasta que el RTC esté marcando la fecha y hora correcta se programa el PIC con el código del medidor. El código en lenguaje C se presenta en las siguientes líneas:

```
//programarRTC.h

#fuses HS, NOWDT, NOPROTECT, PUT, NOWRT, BROWNOUT
#use fast_io(b)
#use delay (clock = 20000000)
#use i2c(master, sda = pin_c4, scl = pin_c3)
```

```
//Direcciones de los registros del DS1307
#define ds_seconds 0x00
#define ds_minutes 0x01
#define ds_hours 0x02
#define ds_day 0x03
#define ds_date 0x04
#define ds_month 0x05
#define ds_year 0x06
#define ds_control 0x07

//Declaracion de las funciones.
void ds1307_SetTimeDate(void);
```

Figura 2.2.2.1. *Archivo de cabecera para programar el RTC-DS1307.*

La Figura 2.2.2.1 muestra el código en lenguaje C del archivo de cabecera del programa programarRTC. En este archivo se declaran las directivas necesarias para el programa. Así como las constantes que representan las direcciones de cada parámetro del RTC-DS1307. Por último se define la única función utilizada en el archivo programarRTC.c.

```
// programarRTC.c

#include <16F876A.h>
#include "programarRTC.h"

void ds1307_SetTimeDate(void)          //configura la hora y fecha.
                                       /Se debe configurar de forma manual la hora y fecha
                                       //al momento en que se programe el PIC.
{
        //los valores de estas variables hay que ajustarlos de forma manual a la fecha y hora que se
programará el PIC.
        int
seconds=0x05,minutes=0x28,hours=0x11,day=0x07,date=0x19,month=0x05,year=0x12;

        i2c_start();                   // Inicia comunicación i2c
        i2c_write(0xD0);               // Dirección del esclavo (DS1307) en modo escritura
        i2c_write(ds_seconds); // Iniciará en la dirección de registro Seconds
        i2c_write(seconds);            // Configura segundos
        i2c_write(minutes);            // Configura minutos

        i2c_write(hours);              // Configura hora
        i2c_write(day);          // Configura dia de la semana
        i2c_write(date);         // Configura dia del mes
        i2c_write(month);              // Configura mes
        i2c_write(year);         // Configura año
        i2c_stop();                    // Finaliza comunicación i2c
}

void main(void)
  {
                ds1307_SetTimeDate();
  }
```

Figura 2.2.2.2. *Archivo C para programar el RTC-DS1307.*

En la Figura 2.2.2.2 se detalla el código del archivo programarRTC.c. Como se aprecia la hora y fecha se programa de forma manual y es el programador quién deberá tener cuidado con la configuración del RTC. La línea de código de la Figura 2.2.2.3 es la que se modifica con la hora exacta en que se programará el RTC.

```
int seconds=0x05,minutes=0x28,hours=0x11,day=0x07,date=0x19,month=0x05,year=0x12;
```

Figura 2.2.2.3. *Configuración fecha y hora del RTC.*

La fecha y hora que se configure en la Figura 2.2.2.3, es la que el RTC-DS1307 guardará cuando se ejecute el programa al energizar el microcontrolador. Se pueden definir los siguientes pasos para programar el sensor de energía trifásico: **1)** ajustar la hora y fecha en el archivo programarRTC.c con unos minutos de adelanto; **2)** compilar el programa **3)** Descargar el programa en el PIC; **4)** Desenergizar el sensor; **5)** esperar que sea la hora configurada, cuando esto suceda hay que energizar el microcontrolador; **6)** descargar en el PIC el programa del medidor. Con esto el microcontrolador queda programado marcando la hora y fecha correcta y con el programa que hace las mediciones. El código fuente de los programas en lenguaje C se encuentran almacenados en el CD que se incluye en el trabajo de graduación.

2.2.3. Cálculo de energía.

Cada segundo el microcontrolador captura el valor de corriente presente en cada una de las entradas ADC del PIC, la potencia es calculada multiplicando la corriente por el valor constante de voltaje (120V). Con el valor de potencia por fase, es posible calcular el retardo necesario para que la carga consuma un watt hora.

```
retardo[i] =(( 360000/potencia[i])-(2)-(36000/potencia[i]));
```

Figura 2.2.3.1. *Línea de código para calcular el retardo de cada fase.*

La Figura 2.2.3.1 es la línea en lenguaje C que calcula el retardo que cada fase necesita para consumir 1Wh, al ser tres fases se hace uso de un lazo for, cada vez que se activa la interrupción del timer1 se compara si el retardo en las tres fases se ha cumplido, al lograrse el retardo en cualquiera de las tres fases la cantidad de energía acumulada en la fase es aumentada en una unidad.

```
for(i=0; i<3; i++){
        if(retardo[i] == conta_retardo[i]){
        energia[i]++;
        conta_retardo[i] = 0;
        }
}
```

Figura 2.2.3.2. *Código que compara el retardo.*

El código de la Figura 2.2.3.2 muestra la forma en comparar si el retardo de cada fase se ha cumplido, si esto sucede la energía es aumentada, de lo contrario el programa espera una nueva interrupción del timer1 para volver a comparar el retardo.

2.2.4. Formato de datos enviados por el puerto UART.

El circuito medidor es el encargado de capturar y procesar los valores de corriente. Por el puerto UART del PIC se envían los valores de potencia promedio, de energía acumulada por las tres fases, la hora y la fecha. El promedio de potencia se envía cada segundo, la energía acumulada cada cinco minutos, la hora y fecha cada diez minutos, de igual manera como lo hace flukso, el formato en que se envían estos datos se muestra en la Tabla 2.2.4.1:

COMANDO	ID MEDIDOR	MEDICIÓN	FIN	PARÁMETRO
pwr	314f76edc4dcf577fa0a49b8335l3b31	:0000000120	0A	Potencia activa
pls	314f76edc4dcf577fa0a49b8335l3b31	:0000000300	0A	Energía activa
tim	314f76edc4dcf577fa0a49b8335l3b31	:1110011057	0A	Hora y fecha

Tabla 2.2.4.1. *Formato de los datos enviados por el sensor de energía trifásico.*

El comando representa el tipo de dato a enviar por el puerto UART. **pwr** es el promedio de la potencia en las tres fases. **pls** es la sumatoria de la energía acumulada por las tres fases. **tim** es la hora y fecha enviada hacia el router dir-300. Cada medidor es identificado por un ID diferente de 32 caracteres ASCII. Como tercer parámetro se envía el valor de la medición respectivo a cada comando utilizado.

Para poder enviar los datos por puerto UART, en el código del PIC se utiliza la función ***printf***, para el uso de esta función es necesario agregar la directiva **rs232** de la figura 2.2.4.1. **baud** es la tasa de transferencia a usar. **xmit** indica el pin del microcontrolador que es usado para la transmisión de datos. **rcv** es el pin utilizado para recibir datos.

```
#use rs232(baud = 4800, xmit = pin_c6, rcv = pin_c7)
```

Figura 2.2.4.1. *Directiva puerto serie.*

Los datos son enviados en forma de caracteres, las líneas de código usadas para enviar la potencia y energía se muestra en la Figura 2.2.4.2.

```
printf("pwr 31f8cbad4babe36ecc67e8179f6fc231:0%09Lu\n" (potencia[0]+potencia[1]+potencia[2])/3);
printf("pls 31f8cbad4babe36ecc67e8179f6fc231:0%09Lu\n",energia_1 + energia_2 + energia_3);
```

Figura 2.2.4.2. *Envío de datos con función printf.*

Para enviar la hora y fecha se utiliza la función **printHoraFecha**, en esta función la primer sentencia a realizar es obtener la hora y fecha del DS1307 y guardar la información en diferentes variables para después poder enviar cada dato por el puerto UART. La Figura 2.2.4.3 lista las líneas de código de la función.

```
void printHoraFecha(void) {
        GetTimeDate();                                  // Lee la hora y fecha del RTC-DS1307.
        printf("tim 31f8cbad4babe36ecc67e8179f6fc231:"); // Se envía el comando tim y ID del medidor.
        putbcd(anio);                                   //Se envía el año.
        putbcd(mes);                                    //Se envía el mes.
        putbcd(dia);                                    //Se envía el día.
        putbcd(hor);                                    //Se envía la hora.
        putbcd(min);                                    //Se envía los minutos.
        putc('\n');
}
```

Figura 2.2.4.3. *Envío de la hora y fecha por puerto UART.*

Con la función GetTimeDate se leen los registros del DS1307 a través de comunicación serie usando el puerto I2C, el RTC retorna la información en formato BCD [11], para poder enviar cada dato como caracter ASCII se procesa cada 4 bits y luego se envía cada uno por el puerto UART, esto lo hace la función **putbcd**.

2.2.5. Simulación en proteus.

Para probar el buen funcionamiento del código, se simuló en el programa PROTEUS antes de descargarlo en el PIC. En la Figura 2.2.5.1 se muestra el diagrama del circuito simulado.

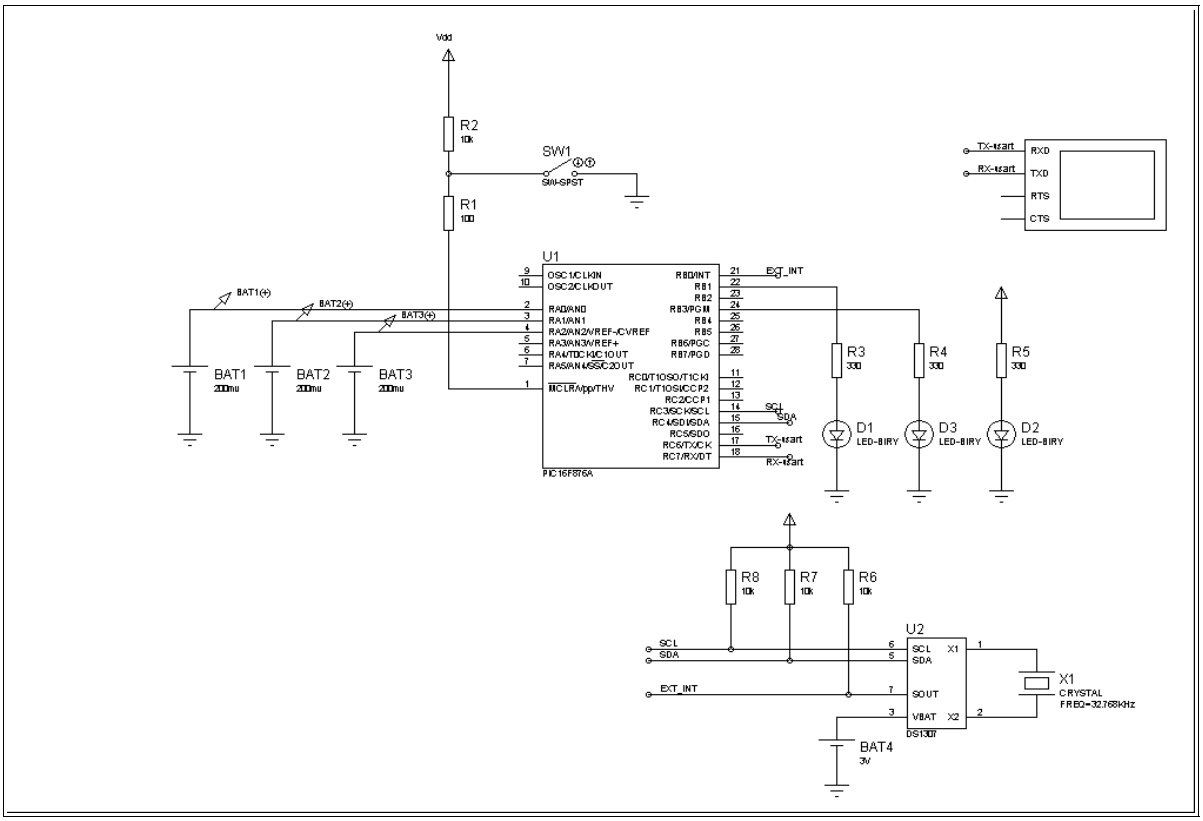


Figura 2.2.5.1. *Circuito simulado en PROTEUS.*

Para simular los transductores de corriente, cada pinza ha sido sustituida por baterias de voltaje directo, estas se han fijado a 200mV para simular una carga de corriente AC de 2Amp; como el voltaje en las tres entradas ADC es el mismo, como resultado se obtiene un promedio de potencia de las tres fases: 2Amp * 120V = 240Watts, el primer dato de energía es enviado a los 5 minutos, por tanto el primer valor enviado al puerto UART tiene que ser: 60Wh, este valor es calculado con las ecuaciones 2.2.5.1 y 2.2.5.2, en la Figura 2.2.5.2 se muestra que la simulación da un valor de 57Wh para los primeros cinco minutos de medición, dato muy cercano al valor esperado.

```
pwr 31f8cbad4babe36ecc67e8179f6fc231:0000000240
pls 31f8cbad4babe36ecc67e8179f6fc231:0000000057
pwr 31f8cbad4babe36ecc67e8179f6fc231:0000000240
pls 31f8cbad4babe36ecc67e8179f6fc231:0000000117
tim 31f8cbad4babe36ecc67e8179f6fc231:1207281442
pwr 31f8cbad4babe36ecc67e8179f6fc231:0000000240
```

Figura 2.2.5.2. *Resultados de la simulación.*

$$Energía_{\ por\ fase} = Pot * Tiempo_{horas} = Pot * (\frac{5\text{min} * 1\text{h}}{60\text{min}}) = 240\text{W} * (\frac{5\text{min} * 1\text{h}}{60\text{min}}) = 20\text{Wh} \quad \text{(ecuación 2.2.5.1)}$$

$$Energía_{total} = 3 * Energía_{\ por\ fase} = 60\text{Wh} \quad \text{(ecuación 2.2.5.2)}$$

Para el manejo del simulador PROTEUS y para aprender a programar microcontroladores en lenguaje C, se han utilizado las bibliografías [1] y [2].

2.3. Diseño con hardware mínimo.

Para lograr reducir los costos del medidor de energía trifásico, se identifica que los bloques: regulador de voltaje 5V y Reloj de tiempo real, son parte del diseño que puede omitirse para lograr reducción de tamaño en la tarjeta impresa, reducción de elementos electrónicos y en general reducción significante en los costos de fabricación del sensor de energía trifásica.

Con el estudio del puerto UART del router DIR-300 [12], se identificó la distribución de pines de la Figura 2.3.1. Donde: **Tx** es el pin de transmisión serie, este es capaz de transmitir hasta un máximo de 57600baudios. **GND** es el punto común del circuito. **Vcc** es salida de alimentación a un voltaje de 3.3V. **NC** es un pin flotante y no tiene conexión con el circuito impreso del router DIR-300. **Rx** es el pin de recepción del puerto UART.

```
Pin1 = Rx; Pin2 = GND; Pin3 = Vcc; Pin4 = NC; Pin5 = Rx
```

Figura 2.3.1. *Distribución de pines J1 DIR-300.*

El Pin3 del J1 tiene presente una fuente de energía de 3.3 voltios, con este valor de energía puede

alimentarse el microcontrolador para su funcionamiento.

Con el objetivo de reducir el espacio utilizado por la batería de litio y los elementos necesarios para que el RTC DS1307 funcione, se optó por instalar un servidor NTP en la computadora configurada como servidor de los servicios web del proyecto.

Al cambiar la fuente de energía que alimenta al microcontrolador, se deberá cambiar el voltaje de referencia del convertidor ADC y por tanto la resolución será como se indica en la ecuación 2.3.1:

$$resolución = \frac{3.3\text{V}}{1024} = 3.22\text{mV} \qquad \text{(ecuación 2.3.1)}$$

Con una carga de 1AAC, se tendrán 100mV-DC en la entrada del ADC, el valor digitalizado por el PIC se calcula usando la ecuación 2.3.2:

$$valorADC = \frac{100\text{mV}}{3.22\text{mV}} = 31.05 \qquad \text{(ecuación 2.3.2)}$$

Para obtener el valor de corriente se usa la ecuación 2.3.3:

$$Corriente = valorADC * (\frac{3.3}{1024}) = 0.1000635\text{Amp} \qquad \text{(ecuación 2.3.3)}$$

Este valor tiene que ser representado como 1Amp, para lograrlo solo falta multiplicar el dato por 10 unidades: $0.1000635 * 10 = 1.000635\text{Amp}$. Como se observa, solo se hace un cambio en el voltaje de referencia positiva del ADC, los datos convertidos por el ADC seguirán siendo válidos.

Con estos cambios realizados, ha sido necesario cambiar el código del microcontrolador, el voltaje de referencia del ADC deja de ser 5V y ahora es de 3.3V, por tanto este cambio hay que indicarlo en las líneas de programación del módulo ADC del microcontrolador. En lenguaje C esto se traduce en usar el valor 3.3 de la figura 2.3.2.

```
corriente = 3.3*readADC/1024;          //Escala de 0 a 3.3 voltios.
```

Figura 2.3.2. *Voltaje de referencia ADC.*

La línea de código de la Figura 2.3.2 usa por defecto el voltaje de alimentación como valores de referencia del ADC. Además como el RTC DS1307 ya no genera una señal de 1Hz, es necesario sustituir la interrupción externa del microcontrolador. Para lograr que el PIC lea las entradas ADC cada segundo, se hace uso de la interrupción timer1, se sabe que esta se activa cada 10ms por lo que se agrega una variable que cuente 100 interrupciones del timer1 equivalente a 1seg para leer los datos del ADC y enviar los datos por puerto UART.

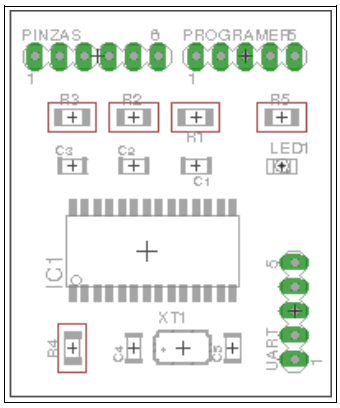


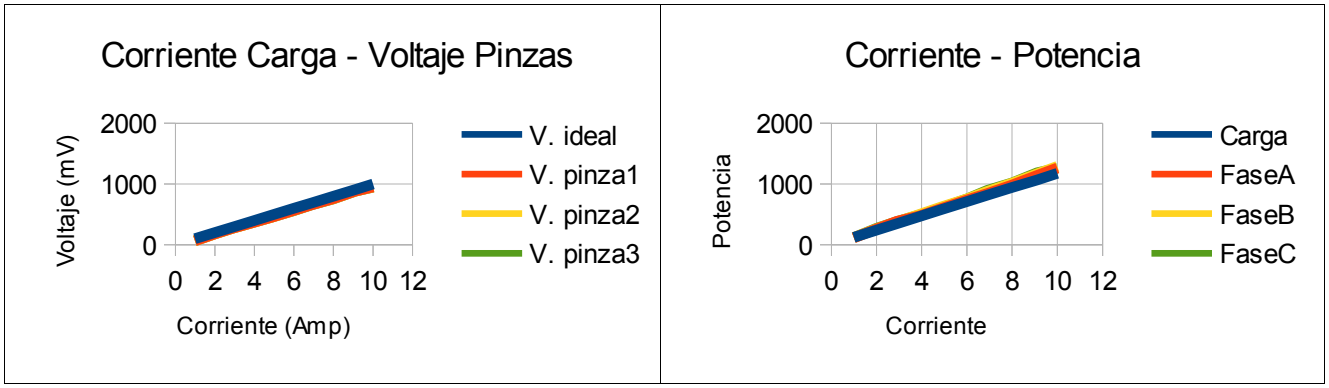
Figura 2.3.3. *Distribución de los elementos en PCB.*

En la Figura 2.3.3 se muestra a manera de ejemplo una distribución en la PCB de los elementos electrónicos, los elementos han sido ubicados en una cara de la tableta, lo que indica que el área total de la PCB se puede seguir reduciendo. Con la distribución de los elementos mostrada en la Figura 2.3.3 el ancho de la PCB es de 3.3cm y el largo de 4cm. Comparando estas medidas con la versión anterior donde el ancho era de 4.5cm y el largo de 6.5cm, la reducción del área de la PCB es del 55%.

2.4. Resultados del circuito medidor.

El sensor medidor de energía envía los datos del PIC a través del puerto UART hacia el router DIR-300. Ejecutando el comando ***cat /dev/ttyS0*** desde la consola de openWRT podemos visualizar los datos que el sensor está enviando. Estos datos son procesados por el router y enviados por medio inalámbrico WIFI hasta el servidor.

Se hicieron mediciones con carga controlada desde 1Amp hasta 10Amp, utilizando focos incandescentes. La Figura 2.4.1.a muestra la gráfica de la corriente AC que circula a través de cada una de las pinzas de corriente, y el voltaje DC medido en cada una de las entradas del sensor de energía. La línea de color azul representa el valor ideal de resolución de las pinzas, con esta gráfica se puede apreciar que cada pinza entrega un valor muy aproximado al valor ideal que el fabricante propone con una resolución de 1AAC/100mVDC.



a) ***b)***

Figura 2.4.1. a) *Corriente/voltaje en las pinzas.* ***b)*** *Corriente de carga/potencia medidor.*

La Figura 2.4.1.b, grafica la corriente que circula a través de cada pinza de corriente, la línea azul representa la potencia consumida por la carga, y las demás líneas es la lectura en el puerto UART obtenida por un medidor; el microcontrolador fué programado de forma exclusiva para obtener cada una de las potencias por fase para el momento en que las mediciones fueron realizadas; en operación normal, el medidor enviará por el puerto UART el promedio de potencia obtenido por las tres fases.

Con los datos obtenidos se calculó el error de la potencia registrada por un medidor, este error varía desde un 3% hasta un 10% del dato real, como el objetivo es mostrar de forma gráfica una tendencia del consumo energético, el porcentaje de error se considera aceptable.

Para hacer funcionar el sensor de energía trifásico, se han consultado y estudiado los archivos de la literatura [5].

Capítulo 3. Medidores de energía en un entorno de red mesh y su configuración.

El presente capítulo describe el diseño de la red utilizada para implementar los medidores de energía eléctrica trifásico funcionando en una topología de red MESH. La red hace uso del protocolo de enrutamiento dinámico ***B.A.T.M.A.N.*** para redes inalámbricas wifi, la red mesh es capaz de encontrar la mejor ruta entre nodos enlazados para poder comunicarse con el SuperNodo. La configuración para esta topología de red es la que utiliza el proyecto DiliVillageTelco [7].

3.1. Diseño de la red.

Se utilizan routers Dlink modelo DIR-300 con firmware openWRT, según su configuración se clasifican como Gateway, SuperNodo y nodosMedidores.

Gateway: provee el servicio de internet al servidor web, el cual es necesario para poder hacer uso del servidor de visualización SPUD. **SuperNodo:** es el encargado de enrutar todo el tráfico de la red mesh hasta el servidor. Cada nodo de la red mesh enruta los datos a través del SuperNodo para que este pueda transferir las mediciones hasta el servidor. **NodoMedidor**: estos registran las mediciones de potencia y energía eléctrica para poder enviar los datos por la red mesh. El sensor de energía está empotrado en el interior de cada uno de los nodos medidores.

Se utiliza una computadora de escritorio como servidor, la cual recibirá los datos y los mostrará en un navegador web que tenga acceso a la red a la que pertenece el servidor.

El diagrama esquemático de la red se ilustra en la figura 3.1.1:

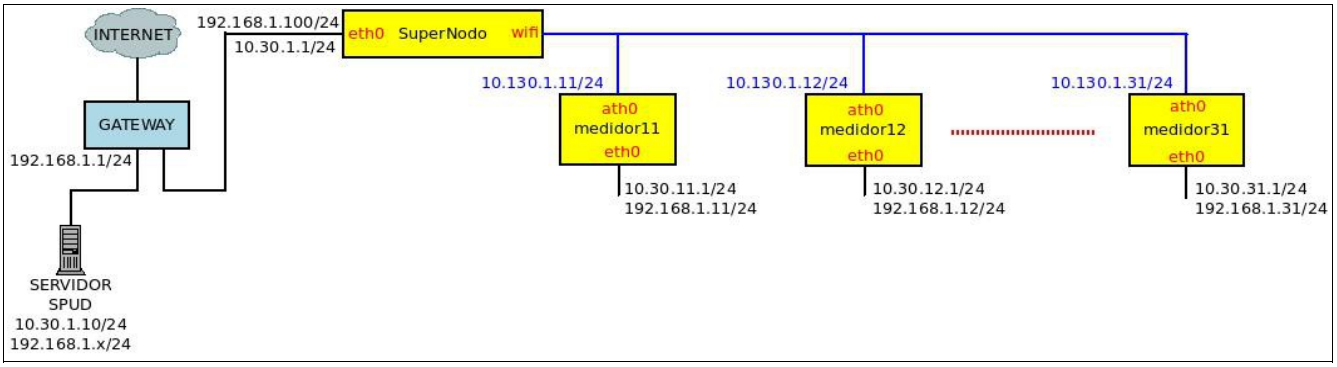


Figura 3.1.1. *Diagrama esquemático de la red.*

La estructura de red inalámbrica WIFI utilizada ha sido configurada tomando de base las bibliografías [6], [7], [8] y [9].

3.2. Configuración de los dispositivos de la red.

A continuación se detallan los pasos necesarios para hacer funcionar la red del diagrama de la figura 3.1.1 y se presentan las configuraciones de cada uno de los nodos.

3.2.1. Configuración de los medidores.

El medidor es un router Dlink DIR-300 con firmware openWRT 10.03 Backfire compilación FLUKSO. Se debe instalar el demonio *B.A.T.M.A.N.D.*; descargando los siguientes archivos desde la página oficial de openWRT [13]: librt_0.9.30.1-43.32_atheros.ipk, libpthread_0.9.30.1-43.32_atheros.ipk, kmod-tun_2.6.30.10-1_atheros.ipk, batmand_r1439-1_atheros.ipk. Todos los archivos están disponibles en el disco que se adjunta con el presente trabajo de graduación.

Usando el comando scp en la carpeta donde se encuentran los archivos descargados, se copian los archivos en la carpeta /tmp del router. La Figura 3.2.1.1 muestra el uso del comando, se listan los archivos a transferir, la dirección IP del router y la carpeta en la que se copiaran los archivo, en este caso /tmp.

```
scp librt_0.9.30.1-43.32_atheros.ipk libpthread_0.9.30.1-43.32_atheros.ipk kmod-tun_2.6.30.10-1_atheros.ipk
batmand_r1439-1_atheros.ipk root@192.168.255.1:/tmp/
```

Figura 3.2.1.1. *Comando para transferir archivos al router.*

Ahora entrar al router via ssh a la dirección IP: 192.168.255.1.

Desde la consola del router hay que cambiarse a la carpeta /tmp; e instalar los paquetes: librt, libpthread, kmod-tun y batmand. Usando el comando opkg install de la manera que lo muestra la Figura 3.2.1.2. El orden mostrado en esta figura debe respetarse para que cada paquete pueda instalarse sin ningún problema de dependencias.

```
root@OpenWrt:/tmp# opkg install librt_0.9.30.1-43.32_atheros.ipk
root@OpenWrt:/tmp# opkg install libpthread_0.9.30.1-43.32_atheros.ipk
root@OpenWrt:/tmp# opkg install kmod-tun_2.6.30.10-1_atheros.ipk
root@OpenWrt:/tmp# opkg install batmand_r1439-1_atheros.ipk
```

Figura 3.2.1.2. *Comando para instalar los paquetes.*

Habilitar batmand usando el comando de la Figura 3.2.1.3, /etc/init.d/ es la dirección en la que se encuentra batmand.

```
root@OpenWrt:~# /etc/init.d/batmand enable
```

Figura 3.2.1.3. *Habilitar el demonio batmand.*

Borrar las IP-TABLES, la Figura 3.2.1.4 lista los archivos que se deben borrar para que batmand pueda enrutar los paquetes sin ningún inconveniente. Estos archivos se encuentran en la carpeta: /etc/modules.d/

```
root@OpenWrt:~# uci commit
root@OpenWrt:~# cd /etc/modules.d/
root@OpenWrt:/etc/modules.d# rm -rf 40-ipt-core
root@OpenWrt:/etc/modules.d# rm -rf 41-ipt-conntrack
root@OpenWrt:/etc/modules.d# rm -rf 42-ipt-nat
root@OpenWrt:/etc/modules.d# rm -rf 45-ipt-nathelper
```

Figura 3.2.1.4. *Borrar ip-tables.*

Reiniciar el router, la Figura 3.2.1.5 muestra el comando reboot que sirve para reiniciar el router sin necesidad de desconectar la fuente de energía, se usa el comando exit para liberar la consola de la sesión ssh.

```
root@OpenWrt:/etc/modules.d# reboot; exit
```

Figura 3.2.1.5. *Comando para reiniciar un router.*

Ahora cargar los archivos de configuración: network, wireless y batmand; a manera de ejemplo se configura el nodo 31. En la Figura 3.2.1.6 se detalla la ruta en la que se deben guardar. Estos archivos han sido creados en un editor de texto y se configuraron para trabajar de forma correcta, los archivos están disponibles en el disco que se adjunta al trabajo de graduación.

```
ues@ues-EIE:~/nodo31$ scp network wireless batmand root@192.168.255.1:/etc/config/
```

Figura 3.2.1.6. *Copiar archivos de configuración.*

Entrar via ssh y reiniciar nuevamente.

Ahora entrar con la ip: **root@192.168.1.31**, al digitar el comando ps -x, veremos el proceso batmand ejecutándose.

Como último paso debemos modificar el archivo ***flukso.lua*** ubicado en la carpeta: */usr/share/lua/flukso/*. Para poder indicarle la dirección IP del servidor a la que enviará los datos de las mediciones, esto se hace usando el editor de texto **vi**, En el archivo hay que buscar las siguientes líneas de código mostradas en la Figura 3.2.1.7 y cambiar la dirección IP:

```
local ok, res = xmlrpc.http.call ("http://10.30.1.10/cgi-bin/AddEnergy.cgi","sample.AddEnergy",Device,os.time(),Energy)
-- print (ok)

local ok, res = xmlrpc.http.call ("http://10.30.1.10/cgi-bin/AddWatts.cgi","sample.AddWatts",Device,Tabla[1],Tabla[2],
    Tabla[3],Tabla[4],Tabla[5],Tabla[6],Tabla[7],Tabla[8],Tabla[9],Tabla[10])
-- print (ok)                    -- En el servidor se ejecutara el proceso AddWatts el cual guardara la medicion
```

Figura 3.2.1.7. *Dirección IP del servidor.*

El nodo 31 ha sido configurado, si existen mas nodos configurados en la red malla se visualizan con el comando: **batmand -cd1**.

3.2.2. Configuración del Super Nodo.

Para el SuperNodo también se utilizó un router Dlink modelo DIR-300 con firmware openWRT 10.03.1 BACKFIRE. La red mesh necesita almenos un SuperNodo para poder enrutar los datos de la red hacia el servidor, el SuperNodo no cuenta con la capacidad de medir energía eléctrica, la principal diferencia en la instalación de batmand con un nodo medidor es el uso del servidor de visualización, este servidor vis es el encargado de monitorizar los nodos y la calidad del enlace wifi en cada uno de ellos dentro de la aplicación SPUD.

Los archivos necesarios para instalar BATMAND en el SuperNodo son: librt_0.9.30.1-43.32_atheros.ipk, libpthread_0.9.30.1-43.32_atheros.ipk, kmod-tun_2.6.30.10-1_atheros.ipk, batmand_r1439-1_atheros.ipk, vis_r1439-1_atheros.ipk. Los archivos se pueden descargar desde el sitio web [13]. También están disponibles en el disco que se adjunta con el presente trabajo de graduación

Usando el comando scp en la carpeta donde se encuentran los archivos descargados, se copian en la carpeta /tmp del router. La Figura 3.2.2.1 muestra el uso del comando, se listan los archivos a transferir, la dirección IP del router y la carpeta en la que se copian los archivo, en este caso /tmp.

```
scp librt_0.9.30.1-43.32_atheros.ipk libpthread_0.9.30.1-43.32_atheros.ipk kmod-tun_2.6.30.10-1_atheros.ipk
batmand_r1439-1_atheros.ipk vis_r1439-1_mips.ipk root@192.168.1.1:/tmp/
```

Figura 3.2.2.1. *Comando para transferir archivos al SuperNodo.*

Entrar al router usando el comando **ssh**, dirigirnos a la carpeta /tmp; instalar los paquetes: librt, libpthread, kmod-tun, batmand y vis:

```
root@OpenWrt:/tmp# opkg install librt_0.9.30.1-43.32_atheros.ipk
root@OpenWrt:/tmp# opkg install libpthread_0.9.30.1-43.32_atheros.ipk
root@OpenWrt:/tmp# opkg install kmod-tun_2.6.30.10-1_atheros.ipk
root@OpenWrt:/tmp# opkg install batmand_r1439-1_atheros.ipk
root@OpenWrt:/tmp# opkg install vis_r1439-1_atheros.ipk
```

Figura 3.2.2.2. *Instalación de los paquetes.*

La figura 3.2.2.2 detalla el orden en que se deben instalar cada uno de los paquetes.

Habilitar batmand y vis, la Figura 3.2.2.3 muestra la forma de hacerlo:

```
root@OpenWrt:~# /etc/init.d/batmand enable
root@OpenWrt:~# /etc/init.d/vis enable
```

Figura 3.2.2.3. *Habilitar batmand y vis.*

Borrar las IP-TABLES, la Figura 3.2.2.4 lista los archivos que se deben borrar para que batmand pueda enrutar los paquetes sin ningún inconveniente. Estos archivos se encuentran en la carpeta:

/etc/modules.d/

```
root@OpenWrt:~# uci commit
root@OpenWrt:~# cd /etc/modules.d/
root@OpenWrt:/etc/modules.d# rm -rf 40-ipt-core
root@OpenWrt:/etc/modules.d# rm -rf 41-ipt-conntrack
root@OpenWrt:/etc/modules.d# rm -rf 42-ipt-nat
root@OpenWrt:/etc/modules.d# rm -rf 45-ipt-nathelper
```

Figura 3.2.2.4. *Borrar ip-tables.*

Reiniciar el router, ahora cargar los archivos de configuración disponibles en el CD anexo al trabajo de graduación: network, wireless, batmand y vis:

```
scp network wireless batmand vis root@192.168.1.1:/etc/config/
```

Figura 3.12. *Transferir archivos de configuración.*

Entrar via ssh y reiniciar nuevamente.

Ahora entrar con la ip: ssh root@**192.168.1.100**, al digitar el comando ps -x, podemos ver los procesos ***batmand*** y ***vis*** ejecutandoce.

Con los pasos realizados tenemos configurado el SuperNodo, si hay mas nodos configurados para una red malla, podemos verlos haciendo uso del comando ***batmand -cd1***.

3.2.3. Archivos de configuración.

Los archivos de configuración que se necesitan para que cada router trabaje en una red tipo malla son: network, wireless, batmand y vis; el archivo vis es usado únicamente en el SuperNodo.

3.2.3.1. Configuración del archivo: network

```
config 'interface' 'loopback'                                   (bloque-1)
        option 'ifname' 'lo'
        option 'proto' 'static'
        option 'ipaddr' '127.0.0.1'
        option 'netmask' '255.0.0.0'

config 'interface' 'lan'                                        (bloque-2)
        option 'ifname' 'eth0'
        option 'proto' 'static'
        option 'ipaddr' '192.168.1.100'
        option 'netmask' '255.255.255.0'
        option 'gateway' '192.168.1.1'
        option 'dns' '192.168.1.1'

config  alias                                                   (bloque-3)
        option 'interface' 'lan'
        option 'proto' 'static'
        option 'ipaddr' '10.30.1.1'
        option 'netmask' '255.255.255.0'

config  'interface' 'wifi0'                                     (bloque-4)
        option 'ifname' 'ath0'
        option 'proto' 'static'
        option 'ipaddr' '10.130.1.1'
        option 'netmask' '255.255.255.0'

config  'route'                                                 (bloque-5)
        option  'interface'     'wifi0'
        option  'target'  '10.30.31.0'
        option  'netmask'       '255.255.255.0'
        option  'gateway'       '10.130.1.31'
```

En el bloque-1 se configura una interfaz de red virtual loopback para poder transferir datos hacia el propio dispositivo de red.

En el bloque-2 se configura la interfaz LAN de manera estática, con esta interfaz el nodo puede acceder a la red 192.168.1.0/24.

En el bloque-3 se configura una interfaz virtual que es usada como alias.

En el bloque-4 se configura la interfaz inalámbrica, esta interfaz es la que pertenece a la red malla y usa el protocolo B.A.T.M.A.N., para enrutamiento dinámico inalámbrico.

En el bloque-5 se configuran las diferentes rutas a las que es necesario comunicar cada dispositivo, cada nodo medidor tiene que poder comunicarse con el SuperNodo, por tanto la ruta a agregar es la ruta

de red a la que pertenece del superNodo (10.30.1.0/24); el SuperNodo tiene que poder comunicarse con todos los nodos medidores, por tanto las rutas que se agregan son todas las direcciones de red a la que pertenece cada nodo medidor; por tanto, para el SuperNodo habrán tantos bloques de rutas como cantidad de nodos medidores tengamos en nuestra red malla, si se agrega un medidor nodo32, se tendrán que agregar las siguientes líneas dentro del archivo network:

```
config  'route'
        option  'interface'     'wifi0'
        option  'target'        '10.30.32.0'
        option  'netmask'       '255.255.255.0'
        option  'gateway'       '10.130.1.32'
```

Con esta nueva ruta configurada en el SuperNodo, este será capaz de poder comunicarse con el nodo32.

3.2.3.2. Configuración del archivo: wireless

```
config wifi-device  wifi0
        option type     atheros
        option channel  1

config wifi-iface
        option device   wifi0
        option encryption       none
        option ssid     OpenWrt
        option mode     ahdemo
        option bssid    01:CA:FF:EE:BA:BE
        option swmerge  1
        option bgscan   0
        option network  wifi0
```

La red inalámbrica malla está configurada para trabajar en el canal#1 (2412Mhz). Para que todos los nodos de la red puedan comunicarse, todos deberán estar trabajando en el canal#1. Si por cualquier motivo se necesita trabajar a otra frecuencia, se deberá cambiar el canal de funcionamiento a todos los dispositivos de la red malla.

Todos los nodos de la red malla usan el mismo nombre de identificación (SSID). Para esta red se ha utilizado OpenWrt, además el modo de funcionamiento es del tipo AHDEMO. Esto es igual que el modo AH-DOC, con la diferencia que el nombre SSID es ocultado para para cualquier tarjeta de red inalámbrica instalada en una computadora. La red malla será invisible para cualquier tarjeta en modo STA.

Para que la red malla funcione, se necesita una identificación de célula (BSSID), esto tiene que ser el mismo para todos los nodos, el ID utilizado es el mismo que se utilizó en las configuraciones del proyecto “Dili Village Telco”.

3.2.3.3. Configuración del archivo: batmand

```
config batmand general
        option interface                ath0
        option announce                 10.30.31.0/24
        option gateway_class
        option originator_interval
        option preferred_gateway
        option routing_class
        option visualisation_srv        10.130.1.1
        option policy_routing_script
```

Se utiliza la interfaz “ath0” para indicarle a B.A.T.M.A.N.D., que la interfaz de red a utilizar es la inalámbrica WIFI, así como el servidor de visualización “vis” instalado se encuentra en la dirección del SuperNodo el cual tiene configurado la dirección IP 10.130.1.1.

3.2.3.4. Configuración del archivo: vis

```
config vis general
                option  'interface'             '-j ath0 eth0'
```

El archivo vis es usado únicamente en el SuperNodo. El archivo es editado para agregar al opción : -j ath0 eth0, con esto se le dice al superNodo que todos los nodos de la red malla en la interfaz inalámbrica (ath0) los comunique a la interfaz LAN eth0 para que el servidor de visualización instalado en un computador pueda visualizar cada nodo de la red malla, el servidor de visualización usado en este proyecto es SPUD.

3.2.3. Configuración del GATEWAY.

El gateway es cualquier router WiFi. Este es el encargado de dar acceso a internet al servidor web de la aplicación. Este gateway es el que genera la red 192.168.1.0/24, el cual tiene la dirección IP 192.168.1.1/24. Se usa la configuración por defecto del router.

3.2.4. Configuración del servidor.

El servidor necesita acceder a dos redes diferentes: la primer red es la que provee servicio de internet, la cual es una dirección IP que se configura de forma dinámica por el servidor DHCP del gateway, esta dirección será del rango 192.168.1.X/24. La segunda red es un alias y es la encargada de comunicar el servidor con los medidores de la red mesh, para configurarla se ejecuta el script configEth0.sh.

Las líneas del script se muestran a en la Figura 3.2.4.1:

```
sudo ifconfig eth0:1 10.30.1.10/24 up

sudo route add -net 10.130.1.0/24 gw 10.30.1.1

sudo route add -net 10.30.11.0/24 gw 10.30.1.1
sudo route add -net 10.30.12.0/24 gw 10.30.1.1
sudo route add -net 10.30.13.0/24 gw 10.30.1.1
sudo route add -net 10.30.14.0/24 gw 10.30.1.1

sudo route add -net 10.30.31.0/24 gw 10.30.1.1
sudo route add -net 10.30.32.0/24 gw 10.30.1.1
sudo route add -net 10.30.33.0/24 gw 10.30.1.1
sudo route add -net 10.30.34.0/24 gw 10.30.1.1
sudo route add -net 10.30.35.0/24 gw 10.30.1.1
sudo route add -net 10.30.36.0/24 gw 10.30.1.1
sudo route add -net 10.30.37.0/24 gw 10.30.1.1
```

Figura 3.2.4.1. *Script para configurar el servidor.*

En la primer línea de la Figura 3.2.4.1 se configura una dirección IP virtual que es usada por el servidor, en esta se usa la misma interfaz LAN física del computador, pero es separada de la red 192.168.1.0/24 de forma lógica. En la segunda línea se configura la ruta para poder acceder a la red inalámbrica mesh. El resto de líneas configuran las rutas para poder llegar a la red LAN de cada uno de los nodos.

3.3. Pruebas realizadas de la red.

Desde una computadora conectada via LAN al nodo31, se hace pruebas de respuesta a diferentes direcciones IP de la red, en la Figura 3.3.1 se muestran resultados del comando ping:

```
carlos@carlos-VAIO:~$ ping -c 4 10.30.31.2
PING 10.30.31.2 (10.30.151.2) 56(84) bytes of data.
64 bytes from 10.30.31.2: icmp_req=1 ttl=64 time=2.08 ms
64 bytes from 10.30.31.2: icmp_req=2 ttl=64 time=1.73 ms
64 bytes from 10.30.31.2: icmp_req=3 ttl=64 time=1.89 ms
64 bytes from 10.30.31.2: icmp_req=4 ttl=64 time=1.74 ms
--- 10.30.31.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.733/1.865/2.081/0.146 ms

carlos@carlos-VAIO:~$ ping -c 4 10.30.31.1
PING 10.30.151.1 (10.30.151.1) 56(84) bytes of data.
64 bytes from 10.30.31.1: icmp_req=1 ttl=64 time=2.77 ms
64 bytes from 10.30.31.1: icmp_req=2 ttl=64 time=2.87 ms
64 bytes from 10.30.31.1: icmp_req=3 ttl=64 time=2.30 ms
64 bytes from 10.30.31.1: icmp_req=4 ttl=64 time=129 ms
--- 10.30.151.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 2.308/34.359/129.483/54.920 ms

carlos@carlos-VAIO:~$ ping -c 4 10.130.1.31
PING 10.130.1.151 (10.130.1.151) 56(84) bytes of data.
64 bytes from 10.130.1.31: icmp_req=1 ttl=64 time=2.80 ms
64 bytes from 10.130.1.31: icmp_req=2 ttl=64 time=3.09 ms
64 bytes from 10.130.1.31: icmp_req=3 ttl=64 time=2.35 ms
64 bytes from 10.130.1.31: icmp_req=4 ttl=64 time=2.38 ms
--- 10.130.1.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 2.359/2.659/3.098/0.314 ms

carlos@carlos-VAIO:~$ ping -c 4 10.130.1.1
PING 10.130.1.1 (10.130.1.1) 56(84) bytes of data.
64 bytes from 10.130.1.1: icmp_req=1 ttl=63 time=4.25 ms
64 bytes from 10.130.1.1: icmp_req=2 ttl=63 time=3.78 ms
64 bytes from 10.130.1.1: icmp_req=3 ttl=63 time=3.36 ms
64 bytes from 10.130.1.1: icmp_req=4 ttl=63 time=3.38 ms
--- 10.130.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 3.366/3.696/4.253/0.364 ms

carlos@carlos-VAIO:~$ping -c 4 10.30.1.1
PING 10.30.1.1 (10.30.1.1) 56(84) bytes of data.
64 bytes from 10.30.1.1: icmp_req=1 ttl=63 time=4.43 ms
64 bytes from 10.30.1.1: icmp_req=2 ttl=63 time=3.84 ms
64 bytes from 10.30.1.1: icmp_req=3 ttl=63 time=3.90 ms
64 bytes from 10.30.1.1: icmp_req=4 ttl=63 time=3.40 ms
--- 10.30.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 3.409/3.897/4.430/0.367 ms

carlos@carlos-VAIO:~$ ping -c 4 10.30.1.10
PING 10.30.1.10 (10.30.1.10) 56(84) bytes of data.
64 bytes from 10.30.1.10: icmp_req=1 ttl=62 time=4.32 ms
```

```
64 bytes from 10.30.1.10: icmp_req=2 ttl=62 time=4.15 ms
64 bytes from 10.30.1.10: icmp_req=3 ttl=62 time=6.48 ms
64 bytes from 10.30.1.10: icmp_req=4 ttl=62 time=4.07 ms
--- 10.30.1.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 4.073/4.759/6.481/1.000 ms
carlos@carlos-VAIO:~$
```

Figura 3.3.1. *Pruebas ping desde una IP de un computador hasta el servidor.*

Las pruebas de la Figura 3.3.1 son realizadas desde una computadora que se conecta al nodo31 de la red mesh, la computadora es configurada con la dirección IP 10.30.31.2/24, el comando ping 10.30.31.2 hace una comprobación de la conexión de red a la propia computadora. Con el comando ping 10.30.31.1 se comprueba la conectividad entre la computadora y el nodo31. El comando ping 10.130.1.31 prueba que la computadora puede acceder hasta la dirección IP del nodo31 que pertenece a la red mesh. Haciendo ping 10.130.1.1 se comprueba que es posible comunicarse desde la computadora hasta la interfaz inalámbrica del SuperNodo. El comando ping 10.30.1.1 comprueba la comunicación desde la computadora hasta la interfaz LAN del SuperNodo. Por último se hace la prueba desde la computadora hasta el servidor con el comando ping 10.30.1.10, con estas pruebas se confirma que la ruta para llegar desde la computadora hasta el servidor funciona de forma correcta.

Capítulo 4

Software de monitoreo de la red mesh y visualización de datos

Para este proyecto se usó SPUD como software para el monitoreo de la red tipo malla, ya que recientes experimentos llevados a cabo en la EIE con routers MP01, demostraron su versatilidad al ser de fácil manejo e instalación en el sistema operativo, lo cual le da portabilidad. Además SPUD es la herramienta web usada por Village Telco para monitorizar redes tipo malla, esto le da estabilidad al sistema al haber pasado previamente como todo software por una fase de depuración. También se dispusieron algunas modificaciones a SPUD para poder visualizar las mediciones de energía y potencia de los medidores diseñados.

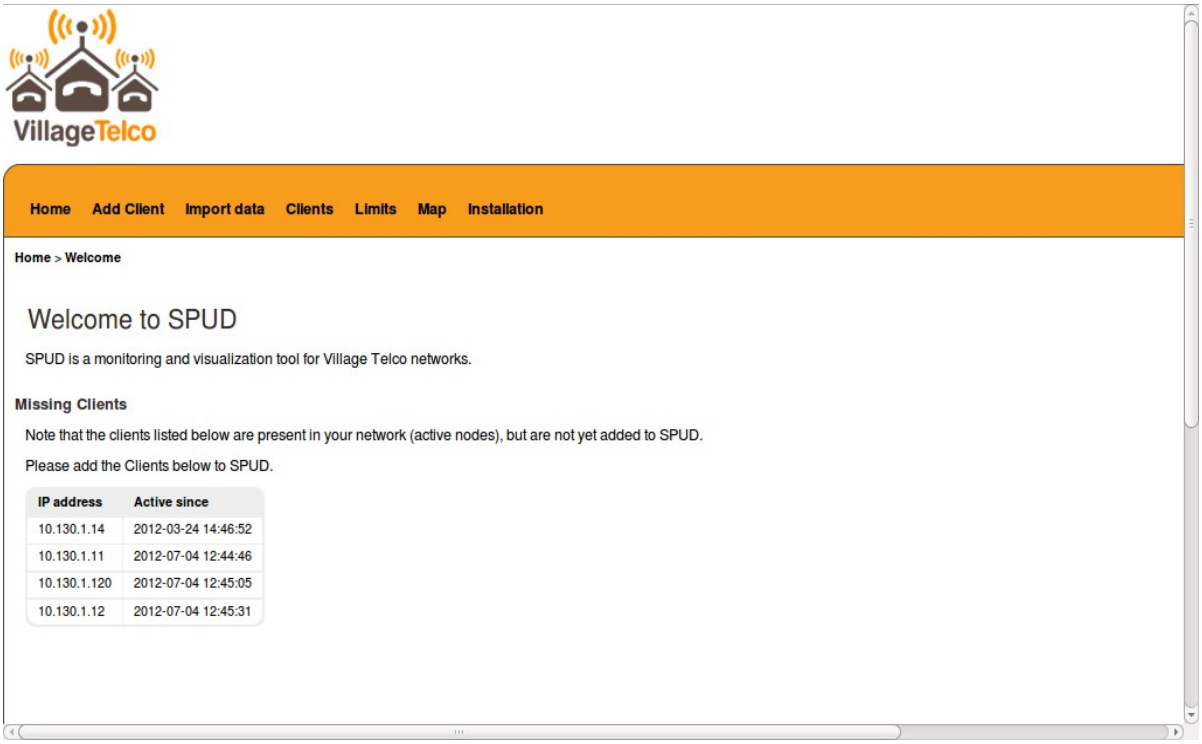


Figura 4.1 *Pagina de inicio SPUD*

SPUD es una interfaz gráfica sencilla implementada en Cake-PHP (una pequeña introduccion a Cake-php se encuentra en el Anexo B) para poder gestionar la red tipo malla. Consta principalmente de tres funciones: añadir nodos, lista de nodos y mapa. La figura 4.1 muestra la pantalla de inicio de SPUD esta se puede acceder desde el navegador web con la URL http://localhost/spud/. Abajo del texto de bienvenida SPUD muestra los nodos que han sido vistos dentro de la red malla que el esta gestionando y aún no han sido dados de alta en el servidor. En la parte superior se encuentra el menú principal se observan las tres funciones mencionadas antes (add client, clients y map) y además incluye una opción poder importar datos de nodos desde un archivo de texto (import data), otra para introducir las métricas de B.A.T.M.A.N (limits), una guía rapida de instalación (instalation) y un enlace a la pagina de inicio

(home). En las siguientes secciones se detallara más que hace cada una de las tres funciones principales de SPUD

Los pasos detallados para la instalación de SPUD (con el cogido fuente modificado) pueden encontrarse en el Anexo C

4.1 Añadir nodo(Add Client)

Este es el primer paso que se debe dar cuando se da de alta a un nuevo miembro en la red de medidores. Esta función sirve para colectar la información de cada uno de los medidores u otro tipo de nodo dentro de la malla. Acá debe introducirse algunos datos requeridos y otros opcionales.

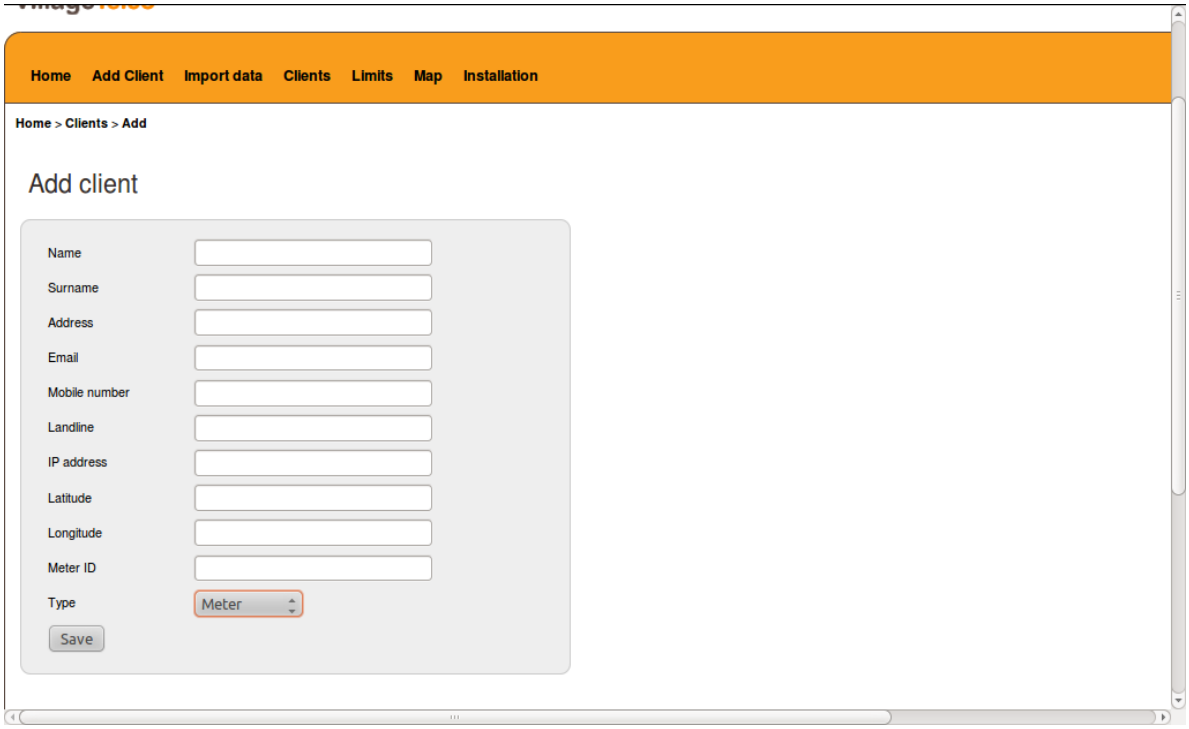


Figura 4.1.1 *Interfaz gráfica de la función Add Client*

La Figura 4.1.1 muestra el formulario requerido para introducir un nodo a la base de datos de SPUD. La información obligatoria para ingresar un nuevo nodo es: el nombre (campo name), dirección ip (campo ip address), coordenadas de localización (campos latitude y longitude) y el tipo de nodo (campo type) este puede ser cliente, súper nodo, gateway o medidor. Los demás campos mostrados en la Figura 4.1.1 no son obligatorios pero sirven para tener mayor información sobre los nodos. Por ejemplo un numero de teléfono (landline), correo electrónico, un número de móvil, etc. Estos campos pueden quedar vacíos. El ID del medidor (campo meter ID) no es requerido por el sistema, pero si se trata de un nodo del tipo medidor este campo debe ser proporcionado. Las mediciones no serán almacenadas si no se incluye el ID del medidor. Mediante modificaciones hechas a SPUD se crea una base de datos RRDTool al momento de introducir el ID del medidor. Al igual que el servidor web implementado en el medidor de energía monofásico[2] (ver Figura 1.5.2) con las modificaciones

hechas a SPUD este tiene dos bases de datos, una implementada en MySQL, donde se almacena la información de los nodos que es introducida en el formulario de la Figura 4.1.1, y otra implementada en RRDTool, que guarda los datos de consumo energético que envían los medidores trifásicos.

4.2 Lista de nodos (Clients)

SPUD incluye la función clients donde se puede ver una lista de todos los miembros de la red con su información.

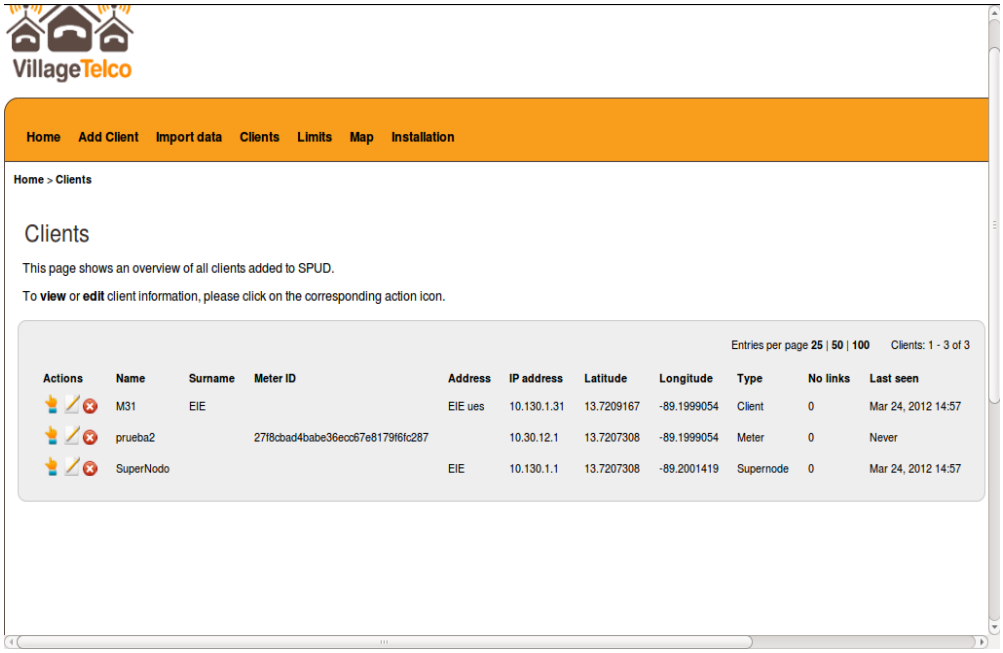


Figura 4.2.1 *Interfaz gráfica de la función Clients*

La Figura 4.2.1 muestra la lista de clientes gestionados por SPUD. La información mostrada por cada nodo corresponde a los datos proporcionados en el formulario para añadir un nodo (ver Figura 4.1.1). Desde la lista de clientes se pueden realizar tres acciones a cualquiera de los nodos para la gestión de la red, estás pueden ser accedidas desde la columna Actions en la lista de nodos de la Figura 4.2.1. La primera acción se representa con el icono de una mano señalando con el dedo índice y sirve para poder ver los detalles del nodo, luego el icono del papel y el lápiz representa la acción editar para poder alterar la información que se proporcionó al ingresar el nodo al sistema; y la ultima acción es para eliminar al nodo del sistema.

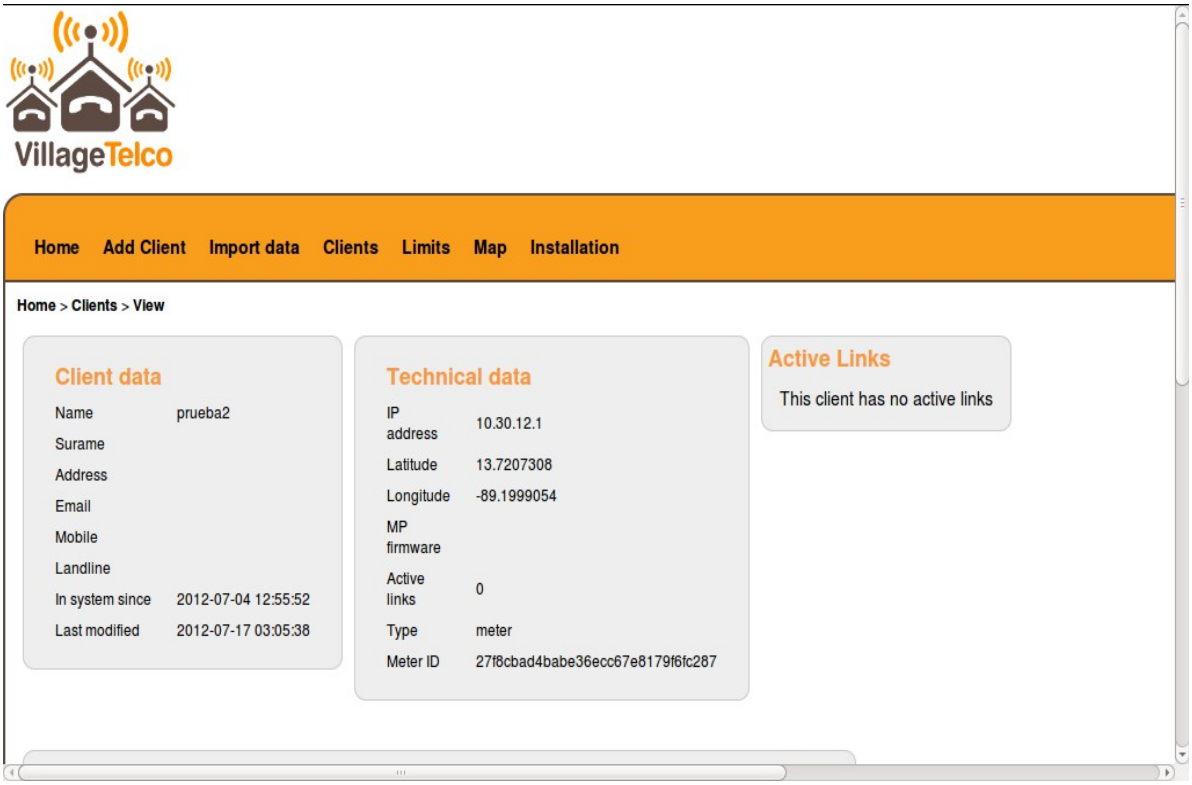


Figura 4.2.2 *Interfaz gráfica de la acción View details*

Los detalles de cada nodo como se aprecia en la Figura 4.2.2 pueden ser accedidos desde la acción ver detalles, en la lista de nodos de la Figura 4.2.1 pinchando el icono del papel y el lápiz en la fila del nodo que se desea consultar, también hay un enlace para cada nodo dentro de la función mapa descrita en la sección 4.3. Los detalles son mostrados en tres recuadros uno que tiene información del nodo (client data) que incluye datos como lo son el nombre, dirección, correo electrónico, etc. Además de la fecha en que se ingresó al sistema y la ultima vez que se modificó. El siguiente recuadro son datos técnicos (Technical data) e incluye la dirección IP, coordenadas de localización, tipo de nodo entre otros. El último recuadro (Active Links) muestra las información sobre los enlaces activos y sus parámetros de red.

Para poder incluir la representación gráfica de los datos enviados por los medidores se incluyo un recuadro más que incluye una imagen generada por RRDTool que puede ser alternada entre gráfico potencia y energía, además de cambiar el eje temporal a intervalos de hora, día, semana, mes o año.

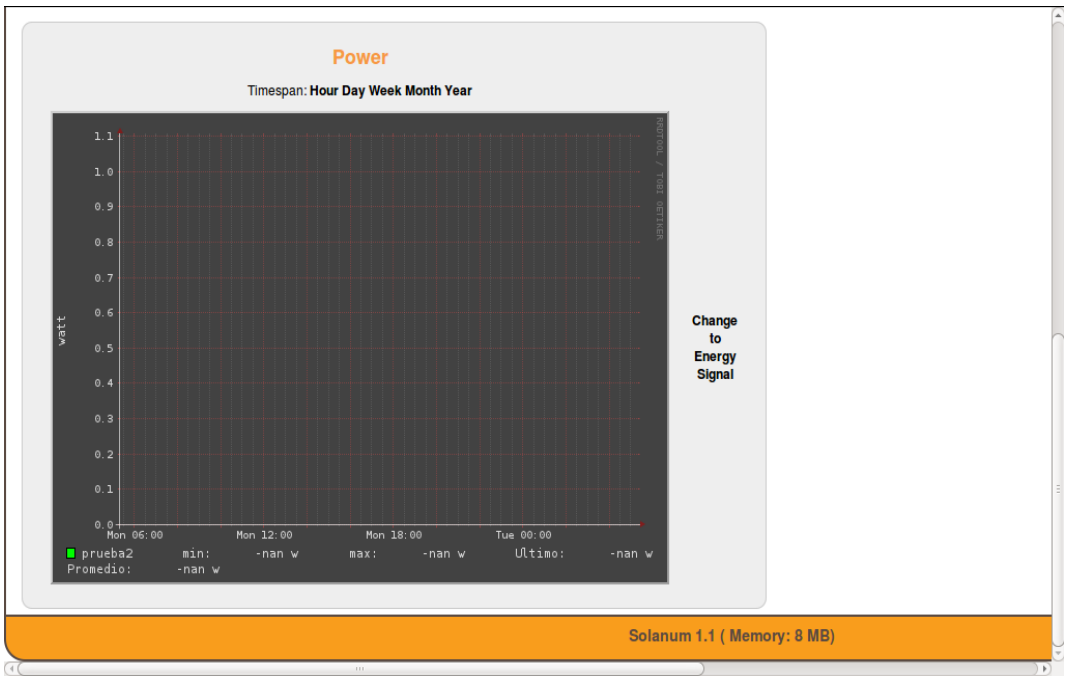


Figura 4.2.3 *Datos de los medidores*

La Figura 4.2.3 muestra el recuadro que se agregó a SPUD para el despliegue de la información de los medidores, en este ejemplo se puede ver una gráfica de potencia aún sin datos, a la derecha se encuentra la opción para cambiar a gráfico de energía y en la parte superior las opciones para el eje temporal.

Al realizar la acción Editar a cualquiera de los nodos con el icono del papel y el lapiz en la Figura 4.2.1, SPUD muestra un formulario con lleno por defecto con la información almacenada en el sistema sobre el nodo, esta puede ser cambiada si es necesario almacenada, además integra un mapa para poder modificar las coordenadas de localización del nodo.

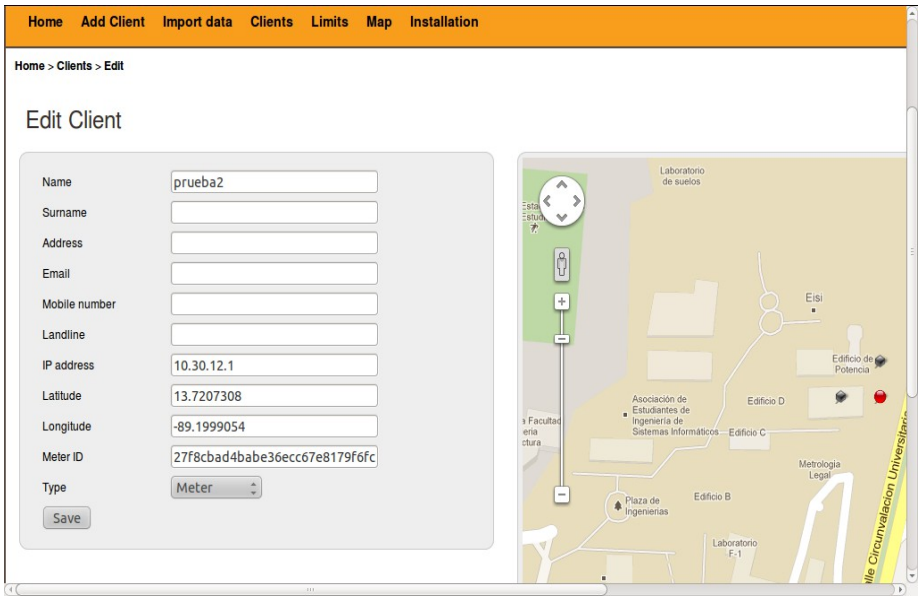


Figura 4.2.4 *Pantalla Editar SPUD*

La Figura 4.2.4 muestra a la izquierda el formulario con la información del nodo y a la derecha el mapa donde pueden cambiarse las coordenadas moviendo el circulo rojo.

4.3 Mapa (Map)

Esta función se utiliza para mostrar visualmente la red. SPUD dibuja un diagrama de la red usando las coordenadas de localización dentro de la información de cada nodo. Para la visualización del mapa SPUD utiliza Google Maps Api 1.3.

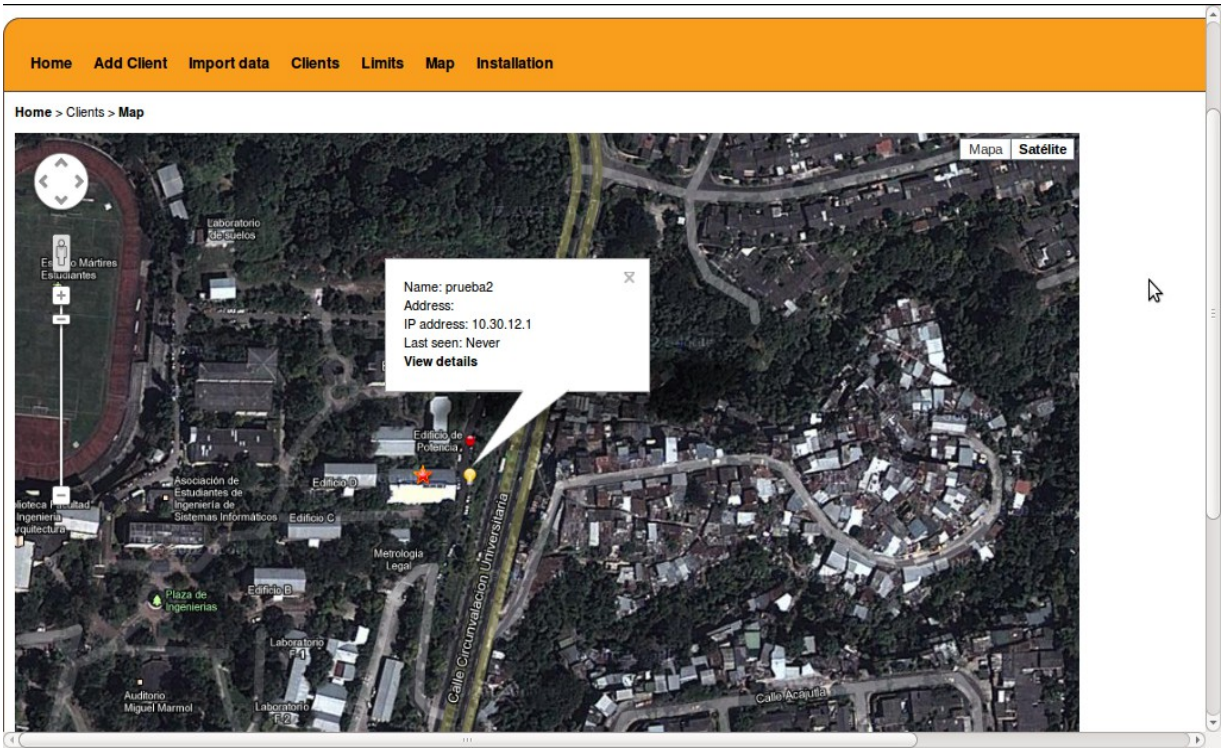


Figura 4.3.1 *Interfaz gráfica mapa*

La Figura 4.3.1 es la interfaz gráfica de SPUD donde se muestra la red sobre un mapa de google maps. Para este ejemplo se muestran tres nodos el icono de la estrella simbolliza un supernodo, el circulo rojo es un gateway. El bombillo muestra un nodo medidor (esta es una de la modificaciones realizadas). Los enlaces se simbolizan por líneas entre los nodos. Si existe una línea verde el enlace es excelente. Si es roja es un enlace pobre. En este ejemplo no se presentan líneas porque las direcciones IP de los nodos no son de la misma red. Por lo tanto no existe comunicación entre ellos y SPUD lo simboliza no mostrando el enlace.

4.4 Modificaciones hechas a SPUD

Para dotar al sistema de las capacidades de mostrar y de almacenar los datos de las mediciones, se realizaron algunas modificaciones al código fuente de SPUD. Esto se logro modificando los scripts relacionados con cada una de las tres funciones principales descritas en los apartados 4.1, 4.2 y 4.3.

Como se mencionó antes el sistema posee dos tipos de bases de datos. Una implementada en MySQL, que donde SPUD almacena originalmente toda la información relacionada a los nodos y los enlaces de red. La otra implementada en RRDTool donde se almacenan las mediciones de Energía y Potencia.

La base de datos MySQL de SPUD posee 4 tablas: *limits, links, mising_nodes y nodes*. Para poder realizar las modificaciones a SPUD solamente se hizo uso de la tabla *nodes*. En esta es donde se almacena toda la información requerida por el sistema cuando se da de alta a un nodo. Como ya se dijo anteriormente esta información se ingresa a través del formulario que se muestra en la Figura 4.1.1. Los campos dentro de la tabla n*odes* son los siguientes: *id, ip_addr, gateway, network, netmask, reated, modified, name, surname, address, mobile. Landline, email, comment, type, lat, long, firmware y last_seen.*

Las bases de datos implementadas para cada medidor en el sistema serán archivos creados con RRDTool con extensión .rrd y con el ID del medidor por nombre.

SPUD está desarrollado en Cake-PHP por lo que a continuación se hará referencia a términos básicos de dicho framework. Se hará el uso de los conceptos: Vista, Modelo y Controlador, debe entenderse por Vista a un archivo escrito en PHP y HTML, Modelo es un archivo escrito en PHP y Controlador también esta escrito en PHP, ver Anexo B para una pequeña introducción a Cake-PHP. Todos los segmentos de código modificados se encuentran plasmados en el Anexo E donde también se incluye la ruta donde esta instalado cada uno de los archivos a los cuales se hace mención en los siguientes apartados

4.3.1 modificaciones en Añadir nodo (Add client)

Las metas perseguidas con las modificaciones hechas a las secciones de código relacionadas a esta función son: incluir un nuevo tipo de nodo llamado medidor, agregar un campo que haga referencia al ID único que identifica a cada medidor y finalmente que el sistema cree automáticamente una base de datos RRDTool al agregar cada nuevo medidor.

Primero se modificó el modelo node (Figura E.1) para describir la forma en que se almacena el identificador de cada medidor, con el propósito de facilitar la tarea se tomó ventaja de un campo llamado comment dentro de la tabla *nodes* en la base de datos MySQL de SPUD. Dentro del modelo node se describe a comment como un campo que no debe estar repetido (no debe permitir medidores con el mismo ID) y como un campo que puede ser de valor null es decir puede estar vacío.

Lo siguiente fue agregar algunas partes de código a la vista add(ver modificaciones en Figura E.2) la cual se encarga de mostrar el formulario que el usuario tendrá que llenar al dar de alta a un nodo por lo tanto debe mostrarse el nuevo tipo de nodo que es medidor y el campo ID del medidor, acá se implemento un Javascript para ocultar la opción para llenar el ID del medidor si se elige un tipo distinto a medidor, para poder incluir el javascript, se modificó la plantilla principal de las vista default.ctp agregando la línea de código que aparece en la Figura 4.3.1.1

```
<?php echo $javascript->link('hidemeterid');?>
```

Figura 4.3.1.1 *Sentencia agregada en default.ctp*

la sentencia en la Figura 4.3.1.1 fue agregada al archivo default.ctp que se encuentra en la ruta /var/www/spud/app/views/layouts/. Para poder lograr la integración con RRDTool creando una base de datos con el nombre del ID del medidor se utilizaron dos scripts desarrollados en el trabajo de graduación Medidor inalámbrico de consumo de energía eléctrica de bajo costo[2], que poseen la función de crear una base de datos RRDTool para las mediciones de energía y otro que hace los mismo para la potencias eléctrica las, bases de datos se alojan en los siguientes directorios dentro del sistema operativo */usr/lib/cgi-bin/rrd_database* para almacenar datos de potencia y */usr/lib/cgi-bin/rrd_database/Energy* para mediciones de energía

para crear bases de datos de potencia se utiliza el script create_rrd.sh alojado en la carpeta */usr/lib/cgi-bin/* (Anexo E Figura E.4) y para las bases de datos de energía se utiliza el script createE_rrd.sh alojado en la carpeta también alojado en el mismo directorio (Anexo E Figura E.5)

Para concluir la integración se añadieron las siguientes lineas de código a la función add dentro del controlador de nodos nodes_controller que cumplen la función de crear las bases de datos en archivos con extensión .rrd y nombre con la cadena de caracteres dentro del campo comment (ID del medidor) además de cambiar los permisos de dichos archivos para que puedan actualizarse periódicamente.

4.3.2 modificaciones en la lista de nodos (Clients)

La primer modificación en esta función de SPUD fue para poder incluir el campo Meter ID en la lista de nodos mostrados en la Figura 4.2.1, estas modificaciones fueron realizadas por la vista de nodos index.ctp (Modificaciones en Figura E.7).

La siguiente modificación es quizás la mas importante o al menos en la que el usuario pasara mayor tiempo si su finalidad es consultar la información sobre los medidores, esta se llevo a cabo en la acción ver detalles (icono del dedo índice en la Figura 4.2.1) el propósito de lo que se hizo en esta sección del código es mostrar de forma gráfica las mediciones almacenadas en las bases de datos RRDTool si los detalles que se están consultando pertenecen a un nodo tipo Medidor, se hizo uso de algunas herramientas ya desarrollados para la primer versión de los medidores [2]estos fueron dos scripts que crean imágenes con las gráficas para mediciones de potencia y energía respectivamente.

En el Anexo E se incluye el código fuente de graficar.sh (Figura E.8) quien se encarga de crear una imagen png con la grafica de potencia y graficarE.sh (Figura E.9) hace los mismo con la Energía, ambos scripts alojados en */usr/lib/cgi-bin/*.

La técnica utilizada para mostrar la gráfica usada en el servidor es sobrescribir una imagen alojada en */var/www/powermeter/graficas_power/* que tendrá por nombre el ID del medidor, con el fin de llevar a cabo esto se modificó la vista details dentro de SPUD (ver Figura E.10 para mas detalles del código)

La ultima modificación hecha a la función lista de nodos lleva la finalidad de mostrar en el formulario de la Figura 4.2.4 de la acción editar, el campo ID del medidor (campo comment en la tabla *nodes* MySQL), lo que se realizo en la vista edit fue similar a lo de la vista add, el archivo de la vista edit es */var/www/spud/app/views/nodes/edit.ctp* las modificaciones a este script pueden encontrarse en la Figura E.11

Dentro de la función edit del controlador de nodos nodes_controller.php se realizaron los siguientes cambios para mantener congruentes el nombre de las bases de datos RRDTool con el campo comment de la tabla *nodes* (meter ID para el usuario), es decir para cambiar el nombre del archivo .rrd de ese medidor si se edita el campo Meter ID en formulario de la Figura 4.2.4 y si este es dejado en blanco el archivo .rrd es borrado. Las modificaciones a la función edit se detallan en la Figura E.12

Se debe ser cauteloso al editar el tipo de nodo ya que si se cambia un medidor a cualquier otro tipo de nodo se perderá la base de datos de mediciones que ha sido almacenada hasta la fecha por ese medidor. Esto también puede ocurrir si al editar el ID de un medidor y dicho campo es dejado en blanco ya que este identificador hace referencia a la base de datos RRDTool donde el dispositivo envía sus mediciones, sin embargo, el sistema es capaz de volver a crear una base de datos nueva RRDTool si se vuelve a escribir un identificador para el medidor. Cuando el tipo cambia de cualquier otro a medidor, se crea una nueva base de datos para las mediciones del respectivo dispositivo.

4.3.3 modificaciones en la función mapa(Map)

Acá las lineas agregadas han sido pocas solo se han hecho para mostrar el icono de un foco en el mapa cuando se tiene un nodo del tipo medidor y también adaptar el hiperenlace de la acción ver detalles a una forma compatible con los cambios realizados al sistema para esto ultimo se agregó las siguiente linea de código a la vista map.ctp, la nueva sentencia se muestra a continuación

```
$inner_html .= "<div><a href='/spud/nodes/details/{$node['Node']['id']}?tg=Power&timespan=d' id='link{$unique_id}'>View details</a></div>";
```

Figura 4.3.3.1 código *agregado a vista map.ctp*

La Figura 4.3.3.1 muestra el fragmento de código agregado en el archivo *var/www/spud/app/views/nodes/map.ctp* donde se editó el hiperenlace que lleva a la acción ver detalles, para poder incluir las variables de sesión que muestran un gráfico de potencia de la ultimo día por defecto, al acceder a la acción ver detalles descrita anteriormente.

Para lograr mostrar el icono se utilizó una imagen alojada dentro de SPUD en el directorio /var/www/spud/app/webroot/img/icons/bulb.ico y se renombro como meter.ico.

se le agregaron las siguientes sentencias

```
        case 'meter':
    $icon = "http://".$spud['host']."/spud/img/icons/meter.png";
    break;
```

Figura 4.3.3.2 *archivo lineas agregadas en formatting.php*

La Figura 4.3.3.2 muestra las sentencias agregadas al archivo formatting.php alojado en /var/www/spud/app/views/helpers/ para poder mostrar el icono meter.ico en el mapa si se trata de un nodo del tipo medidor.

4.4 Comunicación entre el servidor de visualización y los medidores

La comunicación entre los nodos medidores y el servidor de visualización se da usando XML-RPC que es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

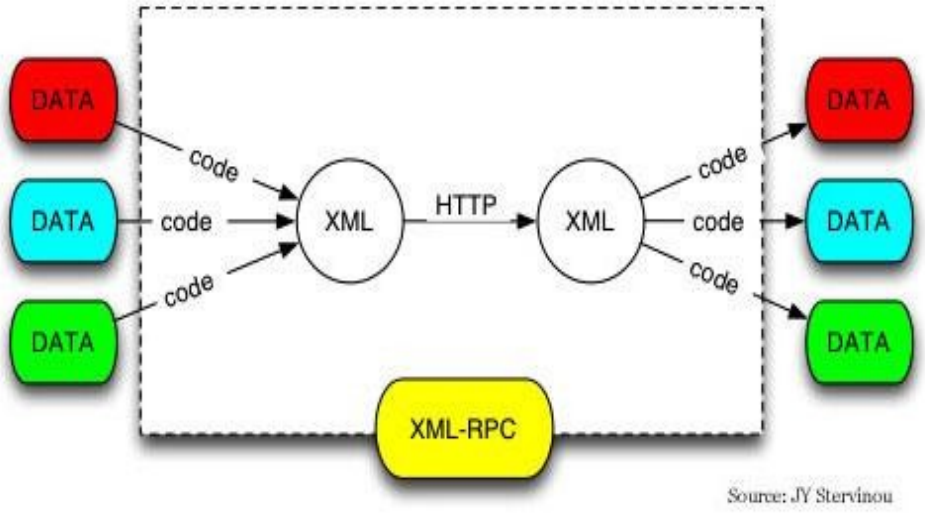


Figura 4.4.1 *Comunicación XML-RPCn [2]*

Desde el script flukso.lua almacenado en cada medidor se hace un llamado remoto a los scripts AddEnergy.cgi y AddWatts.cgi aljados en el directorio /usr/lib/cgi-bin/ dentro del servidor de visualización, estos scripts han sido programados en perl haciendo uso de la librería Frontier::RPC2. Quienes estan encargados de almacenar las mediciones dentro de las base de datos RRDTool de cada medidor.

```
local ok, res = xmlrpc.http.call ("http://10.30.1.10/cgi-bin/AddEnergy.cgi","sample.AddEnergy",Device,os.time(),Energy)
        -- print (ok)

         local ok, res = xmlrpc.http.call ("http://10.30.1.10/cgi-bin/AddWatts.cgi","sample.AddWatts",Device,Tabla[1],Tabla[2],
              Tabla[3],Tabla[4],Tabla[5],Tabla[6],Tabla[7],Tabla[8],Tabla[9],Tabla[10])
        -- print (ok)
```

***Figura* 4.4.2** *fragmento de código de flukso.lua*

La Figura 4.4.2 muestra el segmento donde flukso lua hace un llamado remoto a los scripts AddEnergy.cgi y AddWatts.cgi alojados en el servidor que en este ejemplo en particular tiene la dirección IP 10.30.1.10

AddEnergy.cgi yAddWatts.cgi se muestran en el Anexo E Figura E.13 y E.14

Conclusiones y Líneas a futuro.

1- Los pines del puerto UART del router D'link DIR-300 rotulado como J1 tienen la siguiente distribución: Tx – GND – Vcc – NC – Rx, en el pin Vcc está presente un nivel de voltaje de 3.3VDC, conociento que el microcontrolador puede trabajar en un rango de 2 a 5.5 VDC, se puede pensar en ocupar el pin Vcc del puerto UART para alimentar la circuitería del sensor de energía trifásico, con esto se elimina la necesidad de utilizar el regulador de voltaje REF-195 y todos los elementos asociados a este. Con esto se deberá cambiar el voltaje de referencia del convertidor ADC y por tanto la resolución de acuerdo a la ecuación 2.2.2 será igual a 3.22mV.Con una carga de 1AAC, se tendrán 100mV-DC en la entrada del ADC, por lo tanto el valor digitalizado por el PIC sera 31.05 usando la ecuación 2.2.3. Para obtener el valor de corriente se usa la ecuación 2.2.4 y se obtiene el valor de 0.1000635Amp**.** Este último valor tiene que ser representado como 1Amp, para lograrlo solo falta multiplicar el dato por 10 unidades: 1.000635 Amp esto aproximadamente es 1Amp. Como se observa, solo se hace un cambio en el voltaje de referencia positiva del ADC, los datos convertidos por el ADC seguirán siendo válidos.

2- En el proyecto presentado, se ha incluído el circuito integrado DS1307 que es un Reloj de Tiempo Real, con este se logra marcar la hora y fecha del sistema en todo momento, el uso de una batería de litio tipo pastilla BR-1632A, hace que se use una cantidad considerable de espacio físico en la tableta de circuito impreso; para ahorrar este espacio y poder reducir el tamaño de la placa, se puede retomar la idea de flukso para sincronizar la hora y fecha con los servidores NTP si se tiene acceso de internet. Si no existe acceso a internet puede usarse un servicio NTP instalado en el servidor web.

3- En la configuración de la red mesh no se ha considerado la necesidad de tener acceso a internet en cada medidor. Para lograr una red mesh con mas prestaciones, se puede pensar en configurar los nodos medidores para que puedan dar servicio de internet.

3- El uso de los router para la transmisión de los datos de forma inalámbrica es de gran versatilidad, con esto se da paso a investigaciones con circuitos sensores distintos, por ejemplo podrían diseñarse equipos que midan temperatura, niveles de humedad en el ambiente o cualquier otro instrumento que mida otra variable de interés.

4- Dado que el router Dir-300 ha sido descontinuado por D'Link es necesario que se experimente usando otro tipo de router por ejemplo el router Dragino.

5- Un elemento que eleva los costos de producción de los medidores trifásicos es el uso de las pinzas de Flukso como transductor de corriente, por lo tanto debe investigarse sobre otro tipo de transductor.

6- Una nueva versión del circuito sensor mostrado en la Figura 2.1.5 puede ser desarrollado en líneas futuras de investigación haciendo los cambios propuestos en los numerales 2 y 3, con la eliminación de los elementos relacionados al RTC y a la referencia de voltaje de la alimentación se logra reducir los costos de manufactura, además de la reducción del PCB .

ANEXO A. Importación de los elementos

Todos los elementos necesarios para la manufacturación fueron suministrados por tres proveedores: Todos los dispositivos de soldadura superficial por la empresa Digikey, las tarjetas PCB y conectores por Futurlec y las pinzas de corriente por Flukso. En la Tabla A.1 se incluye los enlaces a cada uno de los elementos dentro de las tiendas en línea de los proveedores.

ELEMENTO	ENLACE TIENDA EN LINEA
REF-195	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=REF195FSZ-ND
PIC-16F876A	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=PIC16F876A-I/SO-ND
DS1307	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=DS1307Z%2B-ND
Batería 3V	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=P291-ND
XT-20Mhz	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=478-4363-1-ND
XT-32.768Khz	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=X1124-ND
LED-rojo	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=160-1167-1-ND
C-220uF	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=338-1789-1-ND
C-100nF (0.1uF)	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=490-1775-1-ND
C-10uF	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=338-1793-1-ND
C-12pF	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=478-1469-1-ND
R-10KOhm	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=RMCF1206JT10K0CT-ND
R-10KOhm	http://www.digikey.com/product-detail/en/ERJ-6GEYJ103V/P10KACT-ND/43118
R-330Ohm	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=RMCF1206FT330RDKR-ND
R-120Ohm	http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=RMCF1206FT120RCT-ND
PCB	http://futurlec.com/PCBService.shtm
Bases Macho 40 pines	http://futurlec.com/Connectors/HEADS40pr.shtml
Bases hembra 40 pines	http://futurlec.com/Connectors/FHEADS40pr.shtml
Pinzas 50A	https://www.flukso.net/shop

Tabla A.1 Enlaces a las tiendas en linea de los proveedores.

Debe tenerse en cuenta que las baterías de litio usadas en el circuito RTC no son enviadas directamente por Digikey ya que la oficina postal de los Estados Unidos prohíbe el envío internacional de paquetes que contengan litio. Para poder importar estas baterías debe de hacerse por medio de un intermediario que las traiga al país.

ANEXO B. Introducción a Cake-PHP

Esta introducción pretende dar el cocimiento básico para entender las modificaciones realizadas a SPUD, si se desea profundizar un poco mas en el tema se recomienda accesar al link en la referencia [10].

Cake-PHP es un marco de desarrollo [framework] rápido para PHP, libre, de código abierto. Se trata de una estructura que sirve de base a los programadores para que éstos puedan crear aplicaciones Web. La lógica de programación usada por Cake-PHP es conocida como MODELO, VISTA, CONTROLADOR

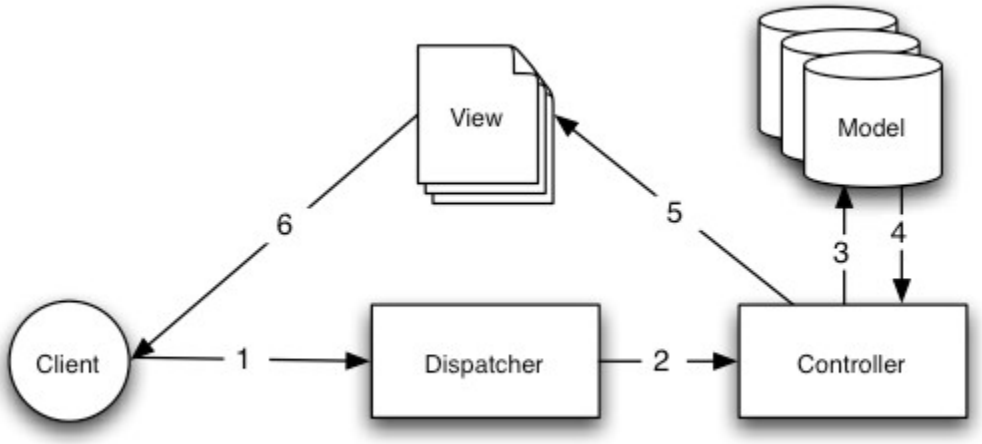


Figura A.1 *Modelo vista Controlador [10].*

Toda interacción con el usuario se hace por medio de las vistas. Las vistas son archivos con extensión ctp (cake template), escritos en lenguaje PHP. Con el uso de las vistas se es capaz de mostrar o solicitar información del usuario final. Esta información sera almacenada en una base de datos o extraída de ella. La base de datos se encuentra modelada en Cake-PHP por los modelos que son una forma de definir que tipo de información será almacenada. Los modelos son archivos que llevan por nombre el singular del nombre de la tabla dentro de MySQL a quien estos representan y con extensión .php. Los modelos están escritos en php. La forma en que se almacena o extrae información de la base de datos se describe en los controladores. Estos son archivos con el nombre del modelo al que manejan seguidos de la siguiente cadena de caracteres *_controller* con la extensión .php. Los controladores están escritos en PHP. Por cada modelo existe un controlador. Cada controlador es una clase que define varias funciones para poder controlar el flujo de información usuario-base de datos. Dentro de cada controlador se encuentran varias funciones dependiendo de las que sean necesarias para poder gestionar la base de datos que ha sido descrita en el modelo al cual el controlador hace referencia.

Rutas de los archivos donde se hicieron modificaciones dentro de SPUD:

Modelos:
/var/www/spud/app/models
Controladores:
/var/www/spud/app/controllers/

Vistas:
/var/www/spud/app/views/

Vistas del modelo nodes
/var/www/spud/app/views/nodes/

Directorio donde se alojan los javascripts
/var/www/spud/app/webroot/js/

Archivo formattin.php
/var/www/spud/app/views/helpers/

Layout por defecto de las vistas default.ctp
/var/www/spud/app/views/layouts/

ANEXO C. Guía para la instalación de SPUD

Para facilitar la instalación de SPUD con las modificaciones realizadas en este trabajo de graduación se implementaron dos script con comandos para el shell con una maquina Ubuntu 11.10. En el CD adjunto a este documento se encuentra un paquete que contiene el sistema SPUDinstall.tar.gz, este paquete descarga las dependencias del sistema y el servicio NTP para medidores que no incluyan el circuito RTC, por lo que al momento de la instalación debe de existir una conexión a internet.

Los pasos para instalar SPUD modificado en una maquina con sistema operativo Ubuntu 11.10, a la cual no tiene instalada la distribución oficial de SPUD, son los siguientes:

1. Descomprimir SPUDinstall.tar.gz en cualquier directorio del sistema operativo.
2. Abrir una terminal de comandos y acceder a la ruta donde se descomprimió el directorio SPUDinstall con la ayuda del comando cd
3. correr el siguiente script instalar con el comando sh instalar.sh
4. Introducir el password del usuario root cuando se necesite
5. Introducir el password que tendra el usuario root de MySQL cuando este sea solicitado.
6. La contraseña por defecto para el usuario root de MySQL incluida en este paquete es 'root' si en el paso 5 se introdujo una distinta, el script conector de Cake-php con MySQL debe modificarse cuando instalar.sh haya terminado de ejecutarse. El archivo es el siguiente : /var/www/spud/app/config/database.php y el password del usuario root de MySQL debe estar escrito en las siguientes lineas de código donde para este ejemplo es '**root'**

```
var $default = array(
                    'driver' => 'mysql',
                    'persistent' => false,
                    'host' => 'localhost',
                    'login' => 'root',
                    'password' => 'root',
                    'database' => 'spud',
                    'prefix' => '',
        );
```

7. El ultimo paso debe hacerse manualemente escribiendo la siguiente sentencia en la linea de comandos: crontab -e cuando se abra el editor de textos se agrega la siguiente linea hasta el final:

```
*/5 * * * * root /usr/bin/wget -O - -q -t 1 http://localhost/spud/nodes/update  >/dev/null 2>&1
```

Los pasos para modificar la versión oficial de SPUD con los cambios hechos en este proyecto en una maquina con Ubuntu 11.10 son los siguientes (SPUD previamente instalado y funcionando correctamente)

1. Descomprimir SPUDinstall.tar.gz en cualquier directorio del sistema operativo.
2. Abrir una terminal de comandos y acceder a la ruta donde se descomprimió el directorio SPUDinstall con la ayuda del comando cd
3. correr el siguiente script parche con el comando sh parche.sh
4. Introducir el password del usuario root cuando se necesite

ANEXO D. Presupuesto

El siguiente presupuesto incluye material para 10 medidores y 4 pinzas de corriente para pruebas

Elemento	Cantidad [UN]	Precio unitario[$]	Precio[$]
PCB doble cara	10	10.23	100.23
Router Dir 300 restaurados	10	15	150
Referencia 5 V REF195	10	3.39	33.9
Microcontrolador PIC 16F876A	10	6.16	61.6
Reloj de tiempo Real DS1307	10	3.35	33.5
Batería con Base 3V	10	1.93	19.3
Cristal 20MHZ	10	1.58	15.8
Cristal 32.768 KHz	10	0.42	4.2
Capacitor 10uF	10	0.44	4.4
Capacitor 220uF	10	0.82	8.2
Capacitor 12pF	20	0.233	4.46
Capacitor 0.1uF	40	0.225	9
Resistencia 120 Ohm	20	0.044	0.88
Resistencia 330 Ohm	30	0.10	3
Resistencia 5.1 KOhm	20	0.06	1.2
Resistencia 100 Ohm	10	0.06	0.6
Resistencia 10 KOhm	10	0.022	0.22
Led rojo	30	0.315	9.45
Bases Hembra 40 pines	4	0.45	1.8
Bases Macho 40 pines	2	0.95	1.9
Pinzas 50 A	4	22.30	89.2
Costos de importación Digikey y futurlec (fletes e impuestos)	1	106.25	106.25
Costos importación Pinzas 50A Flukso (flete, tramite FEDEX e impuestos)	1	118.5	118.5
Total de medidores sin pinzas	10	56.70	569.89
Total 10 medidores y 4 pinzas	$777.59		

Tabla D.1 *Presupuesto para 10 medidores.*

ANEXO E. Modificaciones al código de SPUD

En este Anexo se incluyen los segmentos de código que fueron modificados en SPUD cada uno tiene una pequeña explicación y la ruta del archivo original de SPUD. Cada vez que en las figuras aparecen puntos suspensivos simbolizan el código original.

```
                                                    .
                                                    .
                                                    .
'comment' => array(
                    'isUnque' => array(
                'rule' => array('isUnique'),
                'allowEmpty' =>  true,
                'required'   => false,
                'message' => __('MeterID must be unique.',true)
                )
                                                    .
                                                    .
                                                    .
```

Figura E.1 *Modificaciones hechas en modelo node.php*

La Figura E.1 muestra las lineas de código agregadas al modelo node.php archivo */var/www/spud/app/models/node.php* donde se describe al campo meter ID (comment dentro de MySQL) . Acá se dice que el campo debe ser único para cada registro, que puede estar vacío es decir no es requerido y el mensaje que se mostrara si se trata de almacenar un ID ya existente.

```
$type = array('client'=> __('Client',true),'supernode' => __('Supernode',true), 'gateway' => __('Gateway',true), 'meter'
=> __('Meter',true));
                                                    .
                                                    .
                                                    .

echo "<div  class='frameLeft'><style onload='hidemeterid()'></style>";
                                                    .
                                                    .
                                                    .
array(__("<div id='meterid' style='display:none'>Meter ID</div>",true),          $form-
>input('comment',array('style'=>'display:none','label'=>false))),
    array(__("Type",true),          $form-
>input('type',array('type'=>'select','onchange'=>'hidemeterid()','options'=>$type, 'label'=>false))),
    array($form->submit(__('Save',true),  array('name' =>'data[Submit]', 'class' => 'button')),")
```

Figura E.2 *Modificaciones hechas en vista add.ctp*

Las lineas de código agredas a la vista add.ctp alojada en */var/www/spud/app/views/nodes/* se muestran en La Figura E.2, las primeras sentencias sirven para incluir en la lista de tipo de nodo al tipo con el nombre Meter que tendrá como valor la cadena 'meter' al ser almacenado en el campo type de la tabla nodes en la base de datos de SPUD en MySQL, la segunda sentencia sirve para cargar el script

hidemeterid() y el ultimo bloque sirve para incluir en el formulario el campo Meter ID.

```
function hidemeterid(){
if(document.getElementById('NodeType').selectedIndex==3){
        document.getElementById('NodeComment').style.display='block';
        document.getElementById('meterid').style.display='block';}
else {
        document.getElementById('NodeComment').style.display='none';
        document.getElementById('NodeComment').value='';
        document.getElementById('meterid').style.display='none';}
}
```

Figura E.3 *Javascript /var/www/spud/app/webroot/js/hidemeterid.js*

El Javascript hidemeterid.js mostrado en la Figura E.3 tiene la función de ocultar en el formulario de la Figura 4. y Figura 4. el campo Meter ID cuando una opción distinta a Meter es seleccionada, oculta el texto del formulario "Meter ID" que se encuentra dentro dentro del HTML se encuentra entre una división con la etiqueta meterid y se cambia el estilo a display: none, de manera similar se hace con el recuadro de ingreso de datos que tiene la etiqueta NodeComment.

```
#!/usr/bin/env bash

if [ "$#" != "1" ]; then
        echo "Usage: $0 <database>"
        echo
        echo "Creates an empty rrd database for use with flukso data. Initializes the start time to 30 days ago."
        echo
        echo "Example: $0 test.rrd"
        exit
fi;

PAT="/usr/lib/cgi-bin/rrd_database/$1"

START=`php -r "echo time() - 60*60*24*30;"`
rrdtool create $PAT --start $START -s60 DS:watt:GAUGE:32140800:U:U RRA:AVERAGE:0.5:1:3200
RRA:AVERAGE:0.5:6:3200 RRA:AVERAGE:0.5:36:3200 RRA:AVERAGE:0.5:144:3200 RRA:AVERAGE:0.5:1008:3200
RRA:AVERAGE:0.5:4320:3200 RRA:AVERAGE:0.5:52560:3200 RRA:AVERAGE:0.5:525600:3200
RRA:MIN:0.5:1:3200 RRA:MIN:0.5:6:3200 RRA:MIN:0.5:36:3200 RRA:MIN:0.5:144:3200 RRA:MIN:0.5:1008:3200
RRA:MIN:0.5:4320:3200 RRA:MIN:0.5:52560:3200 RRA:MIN:0.5:525600:3200 RRA:MAX:0.5:1:3200
RRA:MAX:0.5:6:3200 RRA:MAX:0.5:36:3200 RRA:MAX:0.5:144:3200 RRA:MAX:0.5:1008:3200
RRA:MAX:0.5:4320:3200 RRA:MAX:0.5:52560:3200 RRA:MAX:0.5:525600:3200
```

Figura E.4 *Shell Script /usr/lib/cgi-bin/create_rrd.sh*

```
#!/usr/bin/env bash

if [ "$#" != "1" ]; then
        echo "Usage: $0 <database>"
        echo
        echo "Creates an empty rrd database for use with flukso data. Initializes the start time to 30 days ago."
        echo
        echo
"Example: $0 test.rrd"
        exit
fi;

PAT="/usr/lib/cgi-bin/rrd_database/Energy/$1"

START=`php -r "echo time() - 60*60*24*30;"`
rrdtool create $PAT --start $START -s600 DS:watt:GAUGE:32140800:U:U RRA:AVERAGE:0.5:1:3200
RRA:AVERAGE:0.5:6:3200 RRA:AVERAGE:0.5:36:3200 RRA:AVERAGE:0.5:144:3200 RRA:AVERAGE:0.5:1008:3200
RRA:AVERAGE:0.5:4320:3200 RRA:AVERAGE:0.5:52560:3200 RRA:AVERAGE:0.5:525600:3200
RRA:MIN:0.5:1:3200 RRA:MIN:0.5:6:3200 RRA:MIN:0.5:36:3200 RRA:MIN:0.5:144:3200 RRA:MIN:0.5:1008:3200
RRA:MIN:0.5:4320:3200 RRA:MIN:0.5:52560:3200 RRA:MIN:0.5:525600:3200 RRA:MAX:0.5:1:3200
RRA:MAX:0.5:6:3200 RRA:MAX:0.5:36:3200 RRA:MAX:0.5:144:3200 RRA:MAX:0.5:1008:3200
RRA:MAX:0.5:4320:3200 RRA:MAX:0.5:52560:3200 RRA:MAX:0.5:525600:3200
```

Figura E.5 *Shell Script /usr/lib/cgi-bin/createE_rrd.sh*

La Figura E.4 y Figura E.5 muestran los scripts create_rrd.sh y createE_rrd.sh respectivamente, implementados en el trabajo de graduación Medidor inalámbrico de consumo de energía eléctrica de bajo costo[15]. Su forma de uso es desde consola ./sh create_rrd.sh ejemplo.rrd, esta sentencia crearía un archivo con el nombre de ejemplo.rrd en el directorio */usr/lib/cgi-bin/rrd_database.*

```
if($this->data['Node']['type']=="meter"){
                        $meterid=$this->data['Node']['comment'];
                        system("/usr/lib/cgi-bin/create_rrd.sh $meterid.rrd");
                        system("/usr/lib/cgi-bin/createE_rrd.sh $meterid.rrd");
                        system("chmod 777 /usr/lib/cgi-bin/rrd_database/$meterid.rrd");
                        system("chmod 777 /usr/lib/cgi-bin/rrd_database/Energy/$meterid.rrd");
        }
```

Figura E.6 *modificaciones a la función add del controlador nodes_controller.php*

La Figura E.6 muestran el bloque if que fue añadido dentro de la función add del controlador de nodos nodes_controller.sh archivo ubicado en */var/www/spud/app/controllers/nodes_controller.php*, este bloque le dice al controlador de nodos que cuando se añada un nuevo registro a la tabla nodes de la base de datos de SPUD en MySQL, se examine si el tipo es meter y si es correcto se crea una base de datos de energía y potencia y luego se cambian los permisos con chmod para que puedan ser usados por el script AddEnergy.cgi yAddWatts.cgi mostrados en la Figura E.13 y Figura E.14

```
$actions = '<a href="/spud/nodes/details/'.$node['Node']['id'].'?tg=Power&timespan=h" title="Client details"><img
src="/spud/img/icons/view.png"/></a> ';

                                                    .
                                                    .
                                                    .

$row[$key][] = $node['Node']['comment'];
                                                    .
                                                    .
                                                    .

$paginator->sort(__("Meter ID",true), 'Node.comment'),
```

Figura E.7 *sentencias agregadas a index.ctp*

La Figura E.7 muestra las modificaciones realizadas al archivo */var/www/spud/app/views/nodes/index.ctp* con el fin de incluir la columna Meter ID que se muestra en el recuadro de la Figura 4.2.1 La primer sentencia es para el hiperenlace que lleva a la acción ver detalles para éste ejemplo si el nodo es del tipo medidor mostrara por defecto el gráfico de potencia con las mediciones de la ultima hora registrada. La segunda sentencia muestra el campo comment de la tabla *nodes* en MySQL(implementado como ID del Medidor) para cada uno de los nodos y la ultima sentencia imprime el texto Meter ID en la parte superior de la tabla y le da la habilidad de ordenar la lista alfabéticamente de acuerdo al ID del medidor (campo comment en MySQL).

```
#!/usr/bin/env bash

if [ "$#" != "4" ]; then
        echo "Usage: $0 <rrd-database> <output-filename> <Time>"
    echo
        echo "Example: $0 test.rrd output.png h"
        exit
fi;
DB="/usr/lib/cgi-bin/rrd_database/$1"
OUTPUT="/usr/lib/cgi-bin/graficas_Potencia/$2"
FECHA=$3
USUARIO=$4

rrdtool graph $OUTPUT -a PNG --color BACK#424242 --color CANVAS#424242 --color FONT#D8D8D8 --color
AXIS#FFFFFF --vertical-label="watt" --full-size-mode --width 597 --height 424 --alt-autoscale-max --lower-limit=0
--start -1$FECHA \
    DEF:watt=$DB:watt:AVERAGE \
    DEF:wattmin=$DB:watt:MIN \
    DEF:wattmax=$DB:watt:MAX \
    LINE1:watt#00FF00:$USUARIO \
    'GPRINT:watt:MIN:min\: %9.1lf w' \
    'GPRINT:watt:MAX:max\: %9.1lf w' \
    'GPRINT:watt:LAST:Ultimo\: %9.1lf w' \
    'GPRINT:watt:AVERAGE:Promedio\: %9.1lf w' \
     > /dev/null
cp /usr/lib/cgi-bin/graficas_Potencia/$2 /var/www/powermeter/graficas_power/
```

Figura E.8 *shell script graficar.sh*

```
#!/usr/bin/env bash

if [ "$#" != "4" ]; then
        echo "Usage: $0 <rrd-database> <output-filename> <Time>"
      echo
        echo "Example: $0 test.rrd output.png h"
        exit
fi;

DB="/usr/lib/cgi-bin/rrd_database/Energy/$1"
OUTPUT="/usr/lib/cgi-bin/graficas_Potencia/$2"
FECHA=$3
USUARIO=$4

rrdtool graph $OUTPUT -a PNG --color BACK#424242 --color CANVAS#424242 --color FONT#D8D8D8 --color
AXIS#FFFFFF --vertical-label="Wh" --full-size-mode --width 597 --height 424 --alt-autoscale-max --lower-limit=0
--start -1$FECHA \
      DEF:watt=$DB:watt:AVERAGE \
      AREA:watt#21610B:$USUARIO \
      DEF:watt1=$DB:watt:AVERAGE \
      LINE1:watt1#00FF00:$USUARIO \
      'GPRINT:watt:LAST:Ultimo\: %9.1lf wh' \
      'GPRINT:watt:AVERAGE:Promedio\: %9.1lf wh' \
      HRULE:99000#FF0000:"Limite 1 subsidio 99 Kw" \
      HRULE:199000#0000FF:"Limite 2 subsidio 199 Kw" \
       > /dev/null
cp /usr/lib/cgi-bin/graficas_Potencia/$2 /var/www/powermeter/graficas_power/
```

Figura E.9 Shell Script graficarE.sh

Los scripts mostrados en La Figura E.8 y E.9 alojados en */usr/lib/cgi-bin*/ sirven para crear gráficos de potencia (graficar.sh) y energía (graficarE.sh) a partir de una base de datos RRDTool almacenada en */usr/lib/cgi-bin/rrd_database/* para datos de potencia y */usr/lib/cgi-bin/rrd_database/Energy* para energía , la imagen resultante se almacena en */var/www/powermeter/graficas_power/*. Los scripts son corridos desde consola y debe proporcinarsele el nombre la base de datos, el nombre de la imagen que sera creada, el intervalo de tiempo y el nombre del medidor, para ilustrar su uso se dispuso el siguiente ejemplo: ./graficar.sh ejemplo.rrd ejemplo.png h medidorx, la instrucción anterior creara una gráfica de potencia y la almacenara con el nombre de ejemplo.png, a partir de la base de datos ejemplo.rrd la gráfica incluirá los datos de la ultima hora de mediciones y escribira el nombre medidorx en la imagen. Los parámentros aceptados para los intervalos de tiempo son d para datos del ultimo día, w ultima semana, m ultimo mes, y ultimo año y el mostrado en el ejemplo h ultima hora.

```
if($details['Node']['type']=="meter"){
                        $rnd=rand(1,10000);
                        $tg=$_GET['tg'];
                        $timespan=$_GET['timespan'];
                        $meterid=$details['Node']['comment'];
                        $metername=$details['Node']['name'];
                        $nodeid=$details['Node']['id'];
                        if($tg=="Energy"){
```

```
                    system("/usr/lib/cgi-bin/graficarE.sh $meterid.rrd $meterid.png $timespan $metername");
                    $ntg="Power";
                    }
          if($tg=="Power"){
                    system("/usr/lib/cgi-bin/graficar.sh $meterid.rrd $meterid.png $timespan $metername");
                    $ntg="Energy";
                    }
          echo
"<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>";
          echo "<div class='frameLeft'><table class='blue' width='550px'>";
          echo "<tr class='blue'><td align='center'><div class='table_sub_header'>$tg</div></td><tr>";
          echo "<tr class='blue'><td align='center'>Timespan: <b><a href='./$nodeid?
tg=$tg&timespan=h'>Hour </a><a href='./$nodeid?tg=$tg&timespan=d'>Day</a> <a href='./$nodeid?
tg=$tg&timespan=w'> Week</a> <a href='./$nodeid?tg=$tg&timespan=m'>Month</a> <a href='./$nodeid?
tg=$tg&timespan=y'>Year</a></b></td><tr>";
          echo "<tr class='blue'><td align='center'><img src='/powermeter/graficas_power/$meterid.png?
$rnd'></td>";
          echo "<td align='center'><b><a href='./$nodeid?tg=$ntg&timespan=$timespan'>Change to $ntg
Signal</b></tr>";
```

Figura E.10 *líneas de código agregadas a la vista details.ctp*

Las modificaciones realizadas en el archivo */var/www/spud/app/views/nodes/details.ctp* vista para la acción ver detalles se muestran en la Figura E.10, las primeras lineas de código inicializan variables que serán usadas para crear la gráfica, ahí se obtiene con el método get el tipo de gráfico y el intervalo que el usuario desea graficar, también se genera un numero aleatorio que se pasa como parámetro a la imagen para que el navegador en el cliente no almacene en cache y reutilice la imagen que se gráfico anteriormente, después el bloque if ejecuta las sentencias para crear la imagen de la gráfica dependiendo si el usuario desea potencia o energía. Luego las siguientes lineas de código son HTML crudo y se imprimen con desde PHP con echo, estas instrucciones crean una tabla (sin bordes) donde se muestra la imagen de la gráfica y los vínculos para cambiar el tipo de gráfico o el intervalo de tiempo.

```
$type = array('client'=> __('Client',true),'supernode' => __('Supernode',true), 'gateway' => __('Gateway',true), 'meter'
=> __('Meter',true));
                                        .
                                        .
                                        .
echo "<div  class='frameLeft'><style onload='hidemeterid()'></style>";
                                        .
                                        .
                                        .
array(__("<div id='meterid' style='display:none'>Meter ID</div>",true),           $form-
>input('comment',array('style'=>'display:none','label'=>false))),
    array(__("Type",true),           $form-
>input('type',array('type'=>'select','onchange'=>'hidemeterid()','options'=>$type, 'label'=>false))),
    array($form->submit(__('Save',true),  array('name' =>'data[Submit]', 'class' => 'button')),'')
```

Figura E.11 *modificaciones en /var/www/spud/app/views/nodes/edit.ctp*

La Figura E.11 muestra las lineas de código agregadas a la vista edit, es básicamente lo mismo

realizado en la vista add.ctp de la Figura E.2 y tambien se hace uso del script hidemeterid.js de la Figura E.3.

```
$info = $this->Node->find(array('id'=>$id));
        $meterid0=$info['Node']['comment'];
        $meterid1=$this->data['Node']['comment'];
        if ($info['Node']['type']=="meter" and $this->data['Node']['type']!="meter" or $meterid1==null){
                system("rm /usr/lib/cgi-bin/rrd_database/$meterid0.rrd");
                system("rm /usr/lib/cgi-bin/rrd_database/Energy/$meterid0.rrd");
         }
        if ($info['Node']['type']!="meter" and $this->data['Node']['type']=="meter"or $meterid0==null){
                system("/usr/lib/cgi-bin/createE_rrd.sh $meterid1.rrd");
                system("/usr/lib/cgi-bin/create_rrd.sh $meterid1.rrd");
                system("chmod 777 /usr/lib/cgi-bin/rrd_database/$meterid1.rrd");
                system("chmod 777 /usr/lib/cgi-bin/rrd_database/Energy/$meterid1.rrd");
        }
        if ($meterid0!=$meterid1 and $info['Node']['type']=="meter" and $this->data['Node']['type']=="meter"){
                system("mv /usr/lib/cgi-bin/rrd_database/$meterid0.rrd /usr/lib/cgi-bin/rrd_database/$meterid1.rrd");
                system("mv   /usr/lib/cgi-bin/rrd_database/Energy/$meterid0.rrd   /usr/lib/cgi-bin/rrd_database/Energy/
$meterid1.rrd ");
                }
```

Figura E.12 *modificaciones a la función edit del controlador nodes_controller.php*

En la Figura E.12 se encuentra el código agregado a la función edit del archivo */var/www/spud/app/controllers/nodes_controller.php,* Por medio de bloques if se determina si el tipo del nodo que se esta editando es medidor, si es asi si examina si el nuevo valor dentro de meter ID es distinto al valor almacenado previamente, si es distinto se renombra las bases de datos RRDTool de energía y potencia con el nuevo nombre, tambien se examina si este mismo campo fue dejado en blanco y si es asi se borra la base de datos RRDTool del medidor, si por el contrario el campo estaba previamente vacío y se escribe un ID el sistema creara una base de datos nueva RRDTool.

```
#!/usr/bin/perl -w

use strict;
use Frontier::RPC2;


sub AddEnergy {
    my ($Base_rrd1, $x, $y) = @_;

        my @table = @_;
        my $Base_rrd = $table[0];
        my $i;
        my $comp;
        my $Tiempo;
        my $Potencia;

          $Tiempo = $table[1];
          $Potencia = $table[2];

          system("./cargarE.sh $Base_rrd $Tiempo $Potencia");
```

```
    return {'Tiempo' => $x, 'Potencia' => $y, 'BASE'=> $Base_rrd};
}


process_cgi_call({'sample.AddEnergy' => \&AddEnergy});


#===========================================================================
#  CGI Support
#===========================================================================
#  Simple CGI support for Frontier::RPC2. You can copy this into your CGI
#  scripts verbatim, or you can package it into a library.
#  (Based on xmlrpc_cgi.c by Eric Kidd <http://xmlrpc-c.sourceforge.net/>.)

# Process a CGI call.
sub process_cgi_call ($) {
   my ($methods) = @_;

   # Get our CGI request information.
   my $method = $ENV{'REQUEST_METHOD'};
   my $type = $ENV{'CONTENT_TYPE'};
   my $length = $ENV{'CONTENT_LENGTH'};

   # Perform some sanity checks.
   http_error(405, "Method Not Allowed") unless $method eq "POST";
   http_error(400, "Bad Request") unless $type eq "text/xml";
   http_error(411, "Length Required") unless $length > 0;

   # Fetch our body.
   my $body;
   my $count = read STDIN, $body, $length;
   http_error(400, "Bad Request") unless $count == $length;

   # Serve our request.
   my $coder = Frontier::RPC2->new;
   send_xml($coder->serve($body, $methods));
}

# Send an HTTP error and exit.
sub http_error ($$) {
   my ($code, $message) = @_;
   print <<"EOD";
Status: $code $message
Content-type: text/html

<title>$code $message</title>
<h1>$code $message</h1>
<p>Unexpected error processing XML-RPC request.</p>
EOD
   exit 0;
}

# Send an XML document (but don't exit).
sub send_xml ($) {
   my ($xml_string) = @_;
   my $length = length($xml_string);
   print <<"EOD";
Status: 200 OK
Content-type: text/xml
Content-length: $length
```

```
EOD
   # We want precise control over whitespace here.
   print $xml_string;
}
```

Figura E.13 *shell script AddEnergy.cgi*

```
  l#!/usr/bin/perl -w

use strict;
use Frontier::RPC2;


sub AddWatts {
    my ($Base_rrd1, $x, $y) = @_;

        my @table = @_;
        my $Base_rrd = $table[0];
        my $i;
        my $comp;
        my $Tiempo;
        my $Potencia;

       for($i=1; $i<=10; $i++){

           $comp = ($i % 2);
           if($comp != 0 ){
             $Tiempo = $table[$i];
             $Potencia = $table[$i+1];
           }

          system("./cargar.sh $Base_rrd $Tiempo $Potencia");

          }

      return {'Tiempo' => $x, 'Potencia' => $y, 'BASE'=> $Base_rrd};
}


process_cgi_call({'sample.AddWatts' => \&AddWatts});


#============================================================================
#  CGI Support
#============================================================================
#  Simple CGI support for Frontier::RPC2. You can copy this into your CGI
#  scripts verbatim, or you can package it into a library.
#  (Based on xmlrpc_cgi.c by Eric Kidd <http://xmlrpc-c.sourceforge.net/>.)

# Process a CGI call.
sub process_cgi_call ($) {
   my ($methods) = @_;

   # Get our CGI request information.
   my $method = $ENV{'REQUEST_METHOD'};
   my $type = $ENV{'CONTENT_TYPE'};
   my $length = $ENV{'CONTENT_LENGTH'};

   # Perform some sanity checks.
   http_error(405, "Method Not Allowed") unless $method eq "POST";
```

```
    http_error(400, "Bad Request") unless $type eq "text/xml";
    http_error(411, "Length Required") unless $length > 0;

    # Fetch our body.
    my $body;
    my $count = read STDIN, $body, $length;
    http_error(400, "Bad Request") unless $count == $length;

    # Serve our request.
    my $coder = Frontier::RPC2->new;
    send_xml($coder->serve($body, $methods));
}

# Send an HTTP error and exit.
sub http_error ($$) {
    my ($code, $message) = @_;
    print <<"EOD";
Status: $code $message
Content-type: text/html

<title>$code $message</title>
<h1>$code $message</h1>
<p>Unexpected error processing XML-RPC request.</p>
EOD
    exit 0;
}

# Send an XML document (but don't exit).
sub send_xml ($) {
    my ($xml_string) = @_;
    my $length = length($xml_string);
    print <<"EOD";
Status: 200 OK
Content-type: text/xml
Content-length: $length

EOD
    # We want precise control over whitespace here.
    print $xml_string;
}
```

Figura E.14 shell script *AddWatts.cgi*

Las Figuras E.13 y Figura E.14 muestran el código de las rutinas remotas que corren los medidores dentro del servidor web para poder cargar los datos a las bases de datos RRDTool de Energía y potencia ambos archivos se encuentran alojados en */usr/lib/cgi-bin/* y fueron implementados para la monofásica de los medidores de bajo costo [2]. Estos rutinas remotas para conectar con RRDTool usa los shell scripts cargar.sh y cargarE.sh tambien implementados previamente [2] estos no se muestran en este documento pero estan almacenados en el paquete de instalación incluido en el CD adjunto.

REFERENCIAS.

[1] Eduardo García Breijo. Compilador C CCS y simulador PROTEUS para microcontroladores PIC.

[2] Andrés Cánovas López. Manual de usuario del compilador PCW de CCS.

[3] V. García. 64 x 8 Serial Real Time Clock con un ejemplo práctico.

[4] Hoja de datos microcontrolador microchip PIC-16F87X.

[5] Firmware flukso V1: https://github.com/flukso/flm01

[6]Trabajo de graduación EIE UES: Telefonia inalámbrica y red de acceso a internet para los municipios de Salcoatitán, Juayúa, Apaneca y Ataco. Luis Alonso Colocho Susaña y Román Abad Tobías Vides.

[7] http://dili.villagetelco.org/index.php5?title=Main_Page. Página oficial del proyecto Dili Village Telco.

[8] Proyecto de ingeniería EIE UES: Redes Mesh y voz sobre IP. Román Abad Tobias Vides y Rony Stalyn Sánchez Morales.

[9] http://www.open-mesh.org/; protocolo de enrutamiento dinámico B.A.T.M.A.N. para redes mesh wifi.

[10] http://book.cakephp.org/1.3/es/view/879/Comenzando-con-CakePHP. Cake PHP.

[11] http://es.wikipedia.org/wiki/Decimal_codificado_en_binario

[12] http://www.dd-wrt.com/wiki/index.php/Serial_port_pinouts

[13] http://downloads.openwrt.org/backfire/10.03.1/atheros/packages/

[14]Website de Flukso http://www.flukso.net

[15]Trabajo de graduación EIE UES:Medidor inalámbrico de consumo de energía eléctrica de bajo costo. Jonathan Alberto Zaldaña.