UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELÉCTRICA

Optimización del sistema de monitorización remota de medidores de energía eléctrica.

PRESENTADO POR:

ALEXANDER OMAR DUQUE ALAS

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, SEPTIEMBRE 2014

UNIVERSIDAD DE EL SALVADOR

RECTOR :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIA GENERAL :

DRA. ANA LETICIA ZAVALETA DE AMAYA

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERIA ELÉCTRICA

DIRECTOR :

ING. JOSÉ WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE INGENIERIA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

Optimización del sistema de monitorización remota de medidores de energía eléctrica.

Presentado por :

ALEXANDER OMAR DUQUE ALAS

Trabajo de Graduación Aprobado por:

Docente Director :

DR. CARLOS EUGENIO MARTÍNEZ CRUZ

San Salvador, Septiembre 2014

Trabajo de Graduación Aprobado por:

Docente Director :

DR. CARLOS EUGENIO MARTÍNEZ CRUZ

Agradecimientos

Quiero agradecer primeramente a mi Dios padre, a mi Señor Jesucristo y al Espíritu Santo. Porque, a pesar de haber sufrido una enfermedad neuro-cerebral de joven, me ha permitido alcanzar este logro tan importante. Reconozco que me dio una segunda oportunidad de vivir y por su gracia he llegado hasta donde estoy ahora.

Agradezco el apoyo incondicional de mi madre Teresa de Jesús Alas y mi Padre Jaime Omar Duque, quienes han sido un apoyo muy especial durante toda esta jornada. A mi hermano Oscar Armando Duque, quien me ha apoyado a lo largo de toda la carrera.

Agradezco en una manera muy especial a mi hermano Jaime Eduardo Duque. Quien siempre realizo un esfuerzo por apoyarme moral y económicamente. El haber viajado hacia el extranjero, como una persona luchadora en busca de mejores oportunidades me inspiró grandemente a seguir adelante y no darme por vencido, a pesar de las limitantes que se pudieran presentar en el camino.

También quiero agradecer a todo el equipo pastoral de la Iglesia Cristiana Josué Filial Apopa. Quienes me han dado de su confianza para trabajar durante casi 5 años ya dentro del equipo del canal de televisión, Josué TV Canal 46. A mis compañeros de trabajo del canal, a hermanos y amigos dentro de la iglesia, gracias.

Agradezco también al Dr. Carlos Martínez por la oportunidad de trabajar en este proyecto. Por su confianza y apoyo durante el desarrollo de cada una de las etapas de la investigación. Su asesoramiento durante todo el trabajo contribuyo enormemente a que este proyecto haya terminado con éxito.

A todos mis amigos y compañeros de estudio durante toda la carrera, a todos los docentes de la escuela de ingeniería eléctrica.

Gracias.

Alexander Omar Duque Alas

Índice General

Lista de Figuras

Lista de Tablas

Capítulo 1: Introducción

En los últimos diez años el valor de la factura de electricidad del campus central de la Universidad de El Salvador se quintuplicó. En 2001 y 2013, los precios anuales pagados fueron de US$233,008.60 y de US$1,238,128, respectivamente. Ese crecimiento continuo y sostenido se debió a dos razones. La primera, fue el incremento del costo de la energía eléctrica. La segunda, tuvo que ver con el incremento del consumo. Así, por ejemplo, en ese mismo periodo, la carga correspondiente a aire acondicionado se multiplicó por veinte.

En el año 2012, como una forma de determinar, cómo, cuándo y de qué manera se gastaba la energía eléctrica, la Universidad de El Salvador instaló treinta medidores de energía eléctrica. Los modelos adquiridos, aunque del mismo fabricante, contaban con características distintas. Únicamente diez de ellos contaban con capacidad de almacenamiento de datos. Los medidores serían instalados en los postes donde están montadas la mayor parte de las subestaciones del campus. La descarga de datos y su continua monitorización iba a requerir de la exposición del personal al contacto continuo y periódico con las subestaciones.

Como alternativa, a cada medidor se le instaló un *router* inalámbrico tipo *WiFi*. Cada *router* se configuró como un nodo de una red tipo malla, dotando al conjunto de la red de la suficiente redundancia para monitorizar los datos en un punto central. Esa infraestructura de red permitió plantear el desarrollo de aplicaciones de monitorización y de almacenamiento de variables eléctricas.

El trabajo se divide en cuatro partes claramente diferenciadas. La primera corresponde al diseño de una red inalámbrica. Para ello se requiere la configuración de los medidores, la configuración de los *routers* inalámbricos y la integración de todos los equipos en una única red. La segunda parte tiene que ver con la interrogación de los medidores. Para ello se aprovecharon dos cosas. Por una parte, los medidores venían dotados con capacidad de comunicación a través del protocolo MODBUS/TCP. Por otra parte, todos estarían integrados en una única infraestructura de red inalámbrica. La tercera parte tiene que ver con la adecuación del software de monitoreo y la construcción de una base de datos. Por último, la cuarta parte consiste en dotar de capacidades de visualización de datos utilizando herramientas *cloud computing*.

1.1 Necesidad de la investigación

El destinar un computador personal para ejecutar tareas de monitorización continua no solo es un desperdicio de recurso informático sino también de energía eléctrica. Es por ello que este trabajo de graduación es el primero de una serie de trabajos de graduación que buscarán la incorporación de la micro computadora Raspberry Pi. Ésta permite reducir los costos de la monitorización y el almacenamiento de la información.
Además existe la necesidad de resolver los problemas de pérdida de paquetes en la interrogación de los medidores a través de la red inalámbrica. Así como también el envío de esa información a una aplicación web para poder ser monitorizada desde cualquier computadora con acceso a Internet.

1.2. Antecedentes

Este trabajo es la continuación del trabajo de graduación desarrollado en [1]. En la Figura 1, se muestra el diagrama de la red malla implementa.

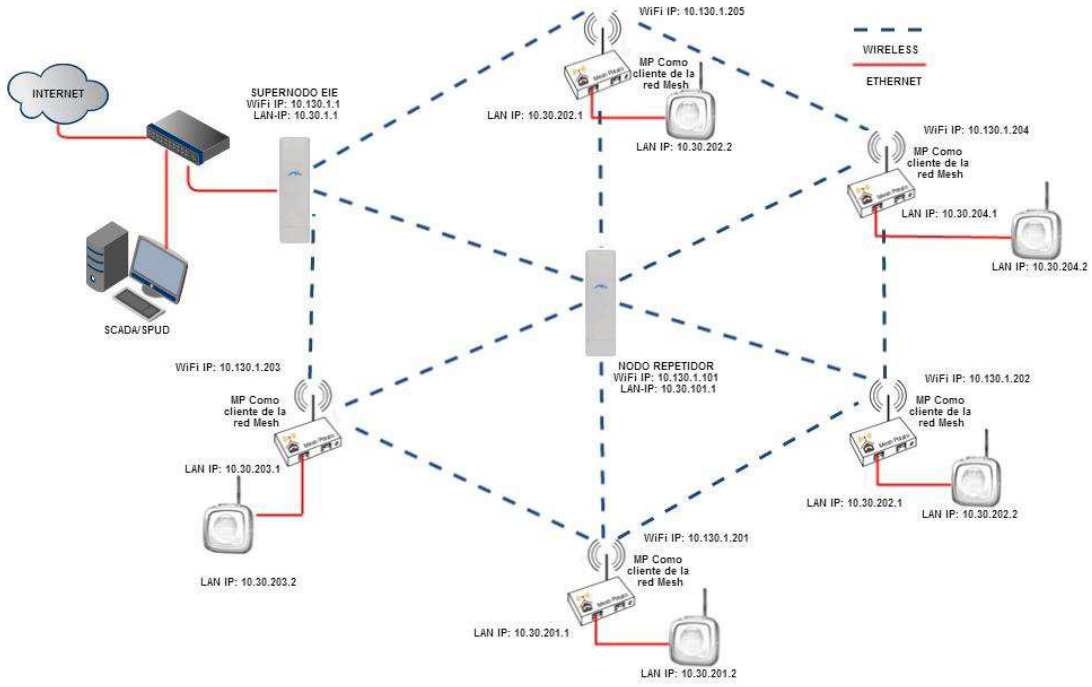


Figura 1. Red de medidores Universidad de El Salvador [1]

Una de las primeras aplicaciones desarrolladas fue la de dotar a la totalidad de medidores la capacidad de almacenamiento. Esto se consiguió mediante un

mecanismo de interrogación continúa. Para ello se aprovechó el que los medidores permitían la comunicación mediante el protocolo MODBUS. La interrogación continúa de los medidores no solo dotó de capacidad de almacenamiento a los medidores que carecen de ella sino qué aumentó la capacidad de almacenamiento de los que ya la tenían.

Esa primera versión del programa de interrogación presentaba diferentes dificultades en la adquisición de datos. Esto debido a que la interrogación se realizaba de manera secuencial. El programa interrogaba a cada medidor uno a la vez. La latencia y los constantes cambios en la calidad de los enlaces de la red inalámbrica hacían que los tiempos de interrogación de cada medidor no fueran uniformes. Además, ese tipo de interrogación imposibilitaba realizar algún tipo de persistencia en la lectura de los medidores, ya que esto aumentaba el tiempo del recorrido de interrogación en toda la red.

La Figura 2. Muestra un esquema sobre cómo se realizaba la interrogación de manera secuencial.

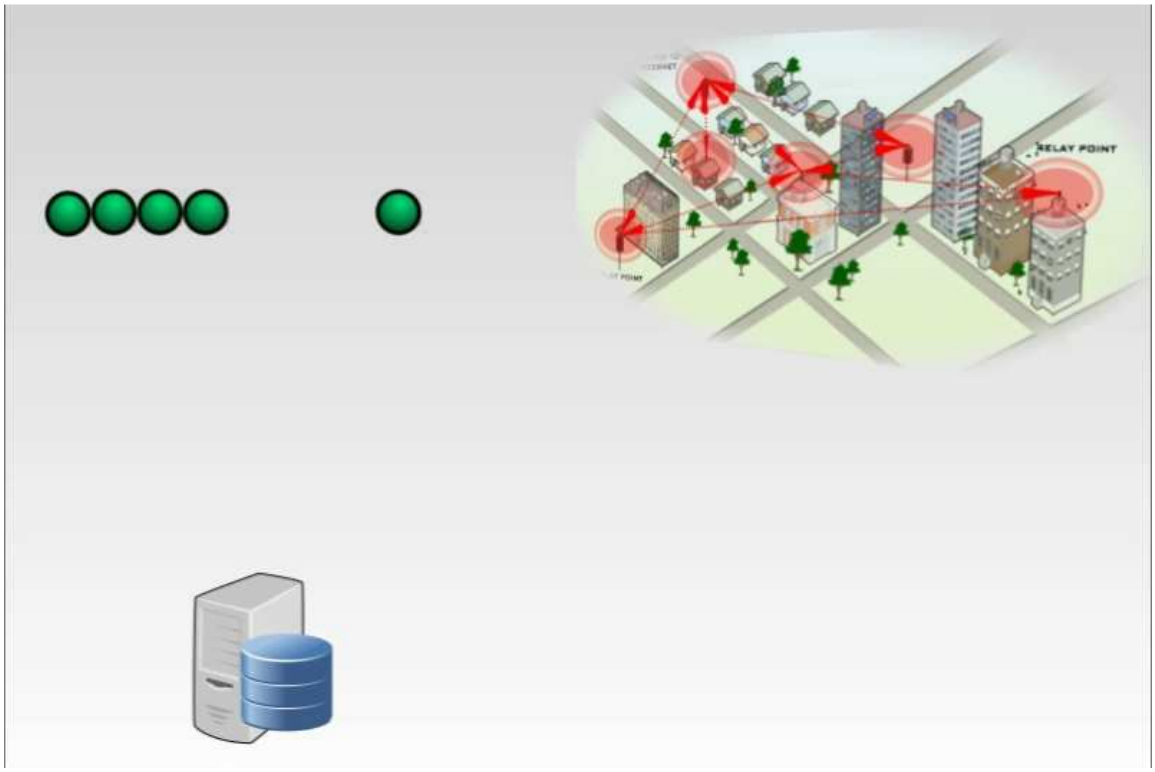


Figura 2. Esquema de interrogación desarrollado en [1]

Cada punto verde representa una interrogación, la cual debe de ir hacia la red y retornar con el valor leído.

En la siguiente subsección se analizará con mayor detalle lo que sucede durante la transacción de los datos. Se hará uso del analizador de protocolos de red *Wireshark* [2]. Esta herramienta cuenta con todas las características estándar de un analizador de protocolos.

1.2.1. Establecimiento de la conexión

Las Figuras 3, 4 y 5 Ilustran como la aplicación Modbus/ TCP establece una conexión TCP. A continuación se explicara en detalle cómo se realiza la conexión entre el medidor (Servidor Modbus) y la computadora (Cliente Modbus). La conexión TCP se realiza en tres pasos:

Paso 1: Como parte de la negociación en tres pasos, el lado cliente de la conexión realiza una apertura activa de un puerto enviando un paquete SYN inicial al servidor (en este caso el medidor). En la Figura 3 se puede observar esta petición, de donde se obtiene la siguiente información:

IP origen (PC): 10.30.1.10
IP destino (medidor): 10.30.206.2
Número de secuencia: 0
Puerto de origen: 60709 (Puertos para usuarios, arriba de 1023)
Puerto de destino: 502 (Puerto del medidor)

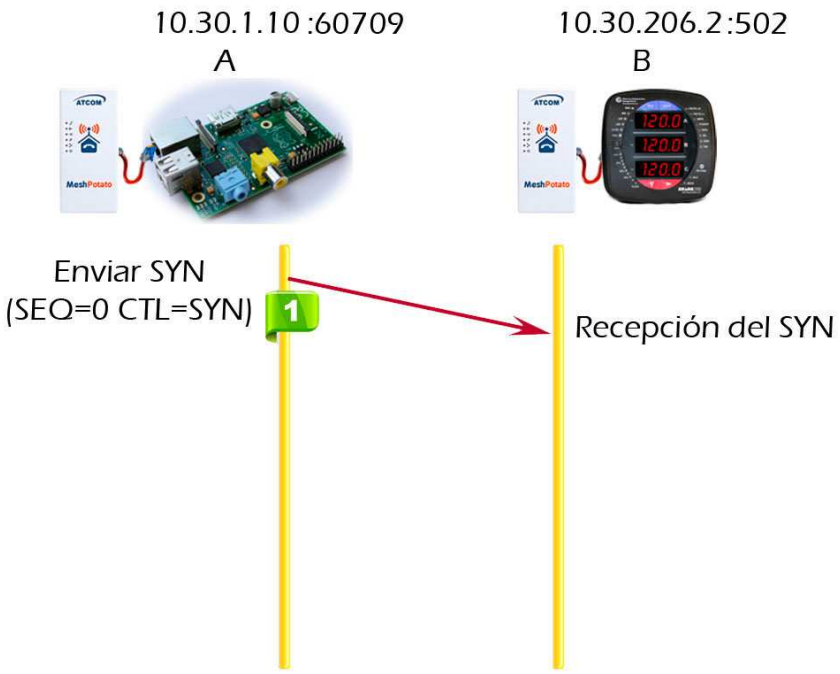


Figura 3. A envía la solicitud de SYN a B.

Paso 2: En el lado del servidor (el medidor) se comprueba si el puerto está abierto, es decir, si existe algún proceso escuchando en ese puerto, pues se debe verificar que el dispositivo de destino tenga este servicio activo y esté aceptando peticiones en el puerto que el cliente intenta usar para la sesión. En caso de no estarlo, se envía al cliente un paquete de respuesta con el bit RST activado, lo que significa el rechazo del intento de conexión. En caso de que sí se encuentre abierto el puerto, el lado servidor respondería a la petición SYN válida con un paquete SYN/ACK. En la Figura 2 se observa este proceso y la información que se obtiene es la siguiente:

IP origen: 10.30.206.2
IP destino: 10.30.1.10
Número de secuencia: 0
Puerto fuente: 502 (Puerto del medidor)
Puerto de destino: 60709 (Puertos para usuarios, arriba de 1023)

En la Figura 4. Se puede observar el diagrama que ilustra el proceso de respuesta ante el paquete SYN

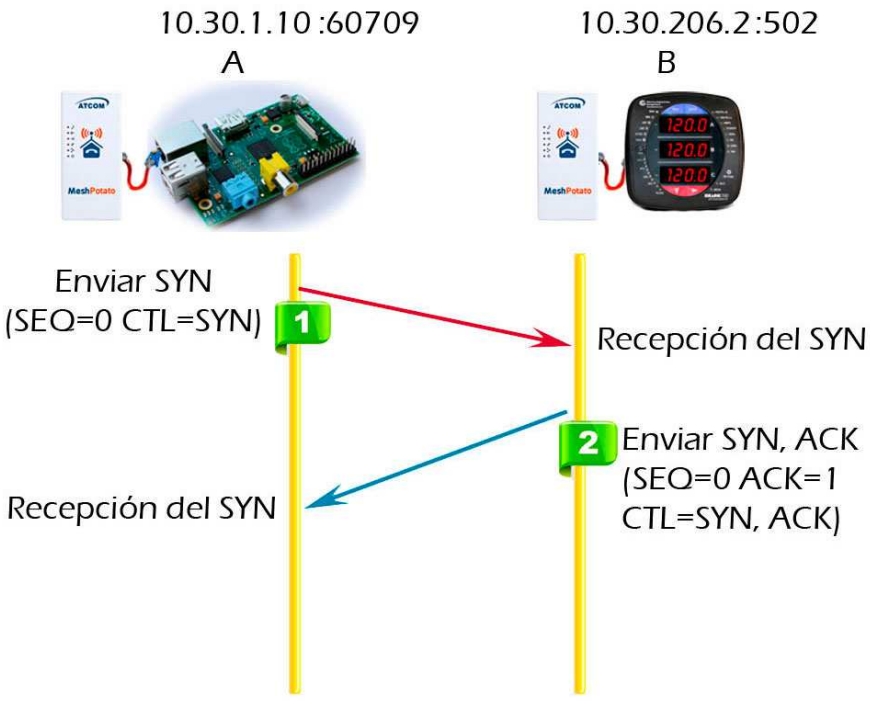


Figura 4. B envía la respuesta de ACK y la solicitud de SYN a A.

Paso 3: Finalmente, el cliente responde al servidor con un ACK, completando así la negociación en tres pasos (SYN, SYN/ACK y ACK) y la fase de establecimiento de conexión. Es interesante notar que existe un número de secuencia generado por cada lado, ayudando de este modo a que no se puedan establecer conexiones falseadas (*spoofing*). En la Figura 5 se observa el diagrama que ilustra la respuesta ACK de la PC donde se muestra la siguiente información:

IP origen (PC): 10.30.1.10
IP destino (medidor): 10.30.206.2
Número de secuencia: 1
Puerto fuente: 60709 (Puertos para usuarios, arriba de 1023)
Puerto de destino: 502 (Puerto del medidor)

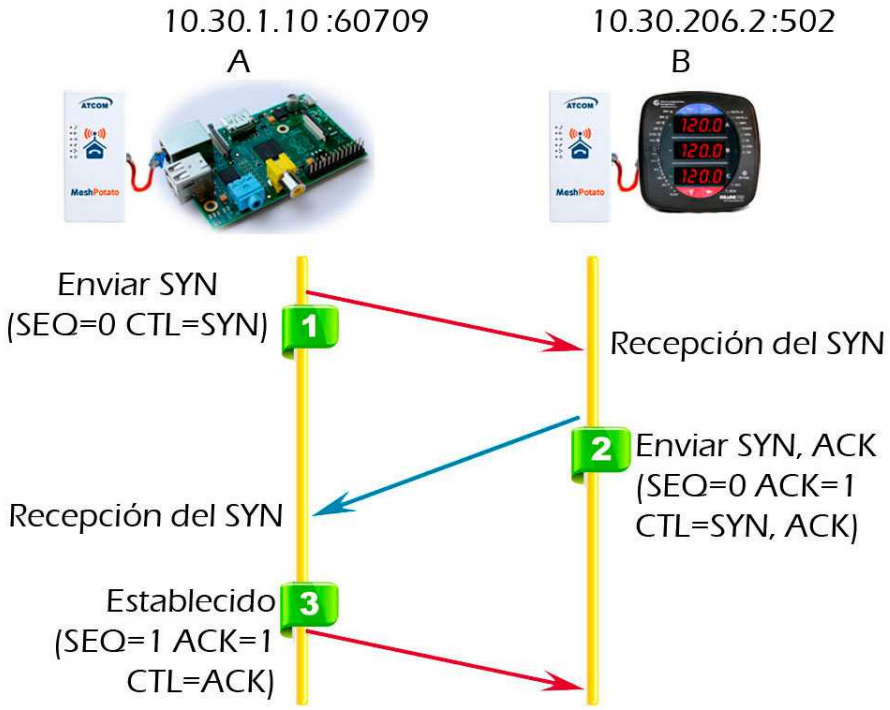


Figura 5. A envía la respuesta de ACK a B.

Como se muestra en la Figura 6, durante la etapa de establecimiento de conexión se identificó un tipo de error en la comunicación de tres pasos. Este error se debe a la debilidad de los enlaces inalámbricos de la red. Una de las claves importantes de este trabajo de graduación fue el corregir este tipo de fallos.

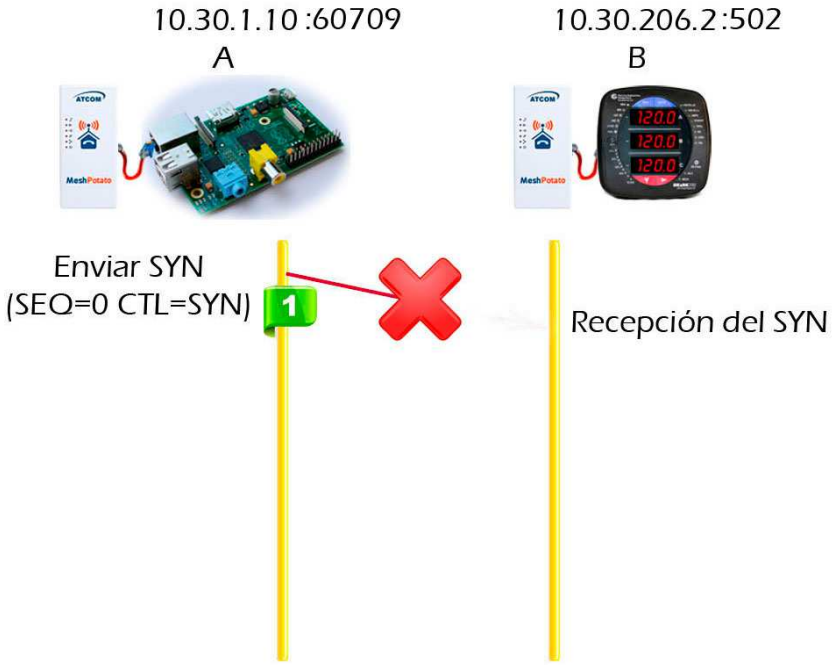


Figura 6. A envía la solicitud de SYN a B.

1.2.2. Transferencia de datos.

El proceso de transferencia de datos se puede dividir en cuatro pasos. El primero es la consulta por parte del cliente, el segundo es la confirmación (ACK) de que se ha recibido la consulta, el tercer paso es el envío de la información por parte del servidor y el último es la confirmación de recibido por parte del cliente. A continuación se describirán con mayor detalle cada uno de estos pasos.

Paso 1: Petición de datos. En la Figura 7(a) se observa el proceso de comunicación TCP. El cliente le hace una petición de datos al medidor. En la parte de datos del segmento TCP viaja la función requerida del servidor. En este caso la función 3. Esta función solicita al servidor que se realice lectura de registros sobre su mapa de memoria. La Figura 7(a) muestra la siguiente información:

IP origen (PC): 10.30.1.10
IP destino (medidor): 10.30.206.2
Función: 3 (En este caso se encarga de hacer la petición)
Puerto fuente: 60709 (Puertos para usuarios, arriba de 1023)
Puerto de destino: 502 (Puerto del medidor)

Paso 2: Confirmación de petición. La Figura 7(a) muestra cuando el medidor responde a la petición del paso anterior con un *ACK.* Para este proceso tenemos la siguiente información.

IP origen: 10.30.206.2
IP destino: 10.30.1.10
Número de secuencia: 1
Puerto fuente: 502 (Puerto del medidor)
Puerto de destino: 60709 (Puertos para usuarios, arriba de 1023)

Paso 3: Envío de información. El medidor envía los datos que se le fueron solicitados en el paso 2. En la Figura 7(a) se observa este envío, de donde se puede obtener la siguiente información:

IP origen: 10.30.206.2
IP destino: 10.30.1.10
Número de secuencia: 1
Código de función: 3 en respuesta (leer registros de retención de salida analógica)
Puerto fuente: 502 (Puerto del medidor)
Puerto de destino: 60709 (Puertos para usuarios, arriba de 1023)

Paso 4: Recepción de información. El cliente envía un ACK como confirmación que recibió los datos. En la Figura 7(a) se observa este envío, de donde se puede obtener la siguiente información:

IP origen (PC): 10.30.1.10
IP destino (medidor): 10.30.206.2
Número de secuencia: 13
Puerto fuente: 60709 (Puertos para usuarios, arriba de 1023)
Puerto de destino: 502 (Puerto del medidor)

Como se muestra en la Figura 7(b), durante la etapa de envío de información se identificó otro tipo de error en la comunicación. De manera similar, este error se debe a la debilidad de los enlaces inalámbricos de la red. Otra de las claves importantes de este trabajo de graduación fue el corregir este otro tipo de fallo.

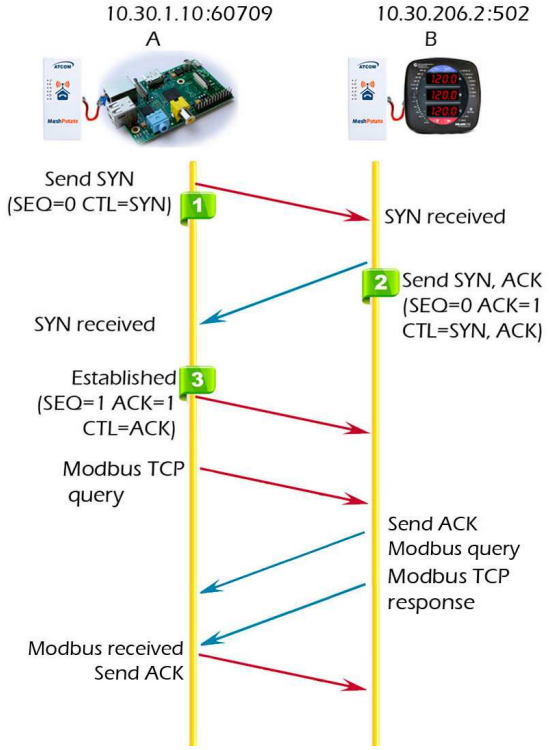
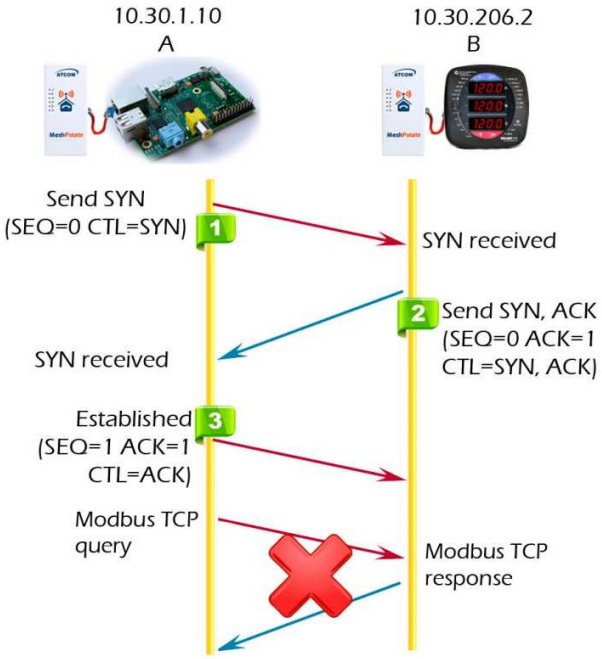


Figura 7(a) Transferencia de datos



7(b) Error en la transferencia de datos.

Los dos tipos de error mostrados anteriormente fueron los problemas a resolver dentro de la red. Para ello se cambió el esquema de interrogación secuencial actualmente en funcionamiento por una interrogación tipo *broadcast*, todos al mismo tiempo y por separado, como se detallará en el siguiente capítulo.

1.3. Motivación para realizar el proyecto.

Este trabajo de graduación pretende añadir más capacidades en cuanto a robustez de la red inalámbrica. Una consecuencia directa de esta mejora será la reducción de perdida de datos. Como se explicó anteriormente esta pérdida de datos se debe a la debilidad de los radio enlaces.

Otra de las motivaciones es la utilización de una microcomputadora como servidor de base de datos. La microcomputadora permitirá la reducción del consumo de energía eléctrica, como también la reducción en gasto del equipo informático.

1.4. Objetivos y organización

1.4.1. Objetivo general

Crear un sistema de monitoreo y almacenamiento de datos de la red de medidores de energía eléctrica del campus central de la Universidad de El Salvador usando la microcomputadora Raspberry Pi.

1.4.2. Objetivos específicos

- Optimizar el software de comunicación, basado en el protocolo Modbus, de la red de medidores de energía eléctrica. Éste debe ser compatible con la microcomputadora Raspberry Pi.

- Instalar y configurar el gestor de base de datos MySQL en la micro computadora Raspberry Pi.

- Integrar la microcomputadora con el software de comunicación Modbus para almacenamiento de datos de la red de medidores.

- Optimizar las rutinas de monitoreo que actualmente están en uso.

- Mejorar la tasa de pérdida de paquetes debido a la variabilidad en la calidad de los enlaces de tipo inalámbricos.

- Crear las funciones para la sincronización del reloj de los medidores shark200s de manera simultánea.

- Enviar datos al sistema de base de datos online (gratuito) de Google App Engine desde la micro computadora Raspberry Pi.

- Crear servicio web usando la plataforma de Google App Engine para mostrar en tiempo real los valores promediados de consumo basados en la normativas de la SIGET (15 minutos) de las mediciones realizadas por la red de medidores de energía.

- Probar el rendimiento del hardware Raspberry Pi. A través de la ejecución continua del software de comunicación Modbus y procesamiento de datos de MySQL, durante un periodo de tiempo de varios meses, se determinará el desempeño de la Raspberry Pi en sistemas de comunicaciones industriales.

1.5 Organización

El trabajo se divide en cuatro partes claramente diferenciadas. La primera corresponde al diseño de una red inalámbrica. Para ello se requiere la configuración de los medidores, la configuración de los *routers* inalámbricos y la integración de todos los equipos en una única red. La segunda parte tiene que ver con la interrogación de los medidores. Para ello se aprovecharon dos cosas. Por una parte, los medidores venían dotados con capacidad de comunicación a través del protocolo MODBUS/TCP. Por otra parte, todos estarían integrados en una única infraestructura de red inalámbrica. La tercera parte tiene que ver con la adecuación del software de monitoreo y la construcción de una base de datos. Por último, la cuarta parte consiste en dotar de capacidades de visualización de datos utilizando herramientas *cloud computing*. Cada una de estas partes se desarrollará a continuación.

Capítulo 2. Interrogación de los medidores

Modbus TCP/IP utiliza la terminología de cliente servidor. Una red Modbus TCP/IP consiste en un cliente conectado a un *switch* o *router* hacia uno o varios dispositivos servidores. Un servidor es análogo a un esclavo en una red Modbus serial. Los dispositivos Modbus TCP/IP utilizan una dirección IP para determinar su ubicación en la red y no un ID como en Modbus RTU (serial).

Como primer paso en la investigación sobre la interrogación de los medidores se realizó una prueba utilizando una computadora, corriendo un programa simulador de medidor. *Diagslave Modbus Slave Simulator* es un pequeño programa de línea de comandos. Es software libre y está disponible para Windows y Linux [2]. Mediante este software se realizaron pruebas de manera segura. Es decir que el desarrollo inicial no requirió el uso de los medidores de energía.

El sistema de direcciones de memoria en el protocolo Modbus está definido por un mapa de memoria dividido en cuatro categorías.

Tipo	Tamaño	Permisos
Bobinas con registros	1 bit	Lectura y Escritura
Entradas discretas	1 bit	Lectura
Registros de entrada	16 bits	Lectura
Registros de escritura	16 bits	Lectura y Escritura

Tabla 1. Sistema de almacenamiento de memoria del protocolo Modbus.

Para valores de coma flotante se requieren registros de 32 bits. El protocolo no especifica cómo se deben implementar registros de memoria para 32 bits. Para poder utilizar registros de 32 bits se utilizan dos registros de 16 bits concatenados. Los medidores interrogados utilizan el estándar IEEE 854-1987 para valores de coma flotante [3].

Los medidores de energía Shark 100-S y Shark 200-S cuentan con la interfaz RS 485 para utilizarse con el modo Modbus RTU y también con un puerto ethernet para utilizar el modo Modbus TCP/IP. Se utilizó el puerto Ethernet, conectándose a un *router* inalámbrico.

El establecimiento de la conexión se simplifica mucho al hacer uso de la librería para lenguaje C Libmodbus, creada por Stéphane Raimbault [4]. Libmodbus es una librería de software libre para enviar/recibir datos a dispositivos usando el protocolo Modbus, disponible para sistema operativo Linux, Windows, Mac OS X, FreeBSD y QNX. Libmodbus permite el desarrollo de aplicaciones open source en lenguaje C para la interrogación de dispositivos.
A continuación la Figura 8 se muestra un diagrama sobre cómo se establece la conexión Modbus TCP/IP usando la librería Libmodbus.

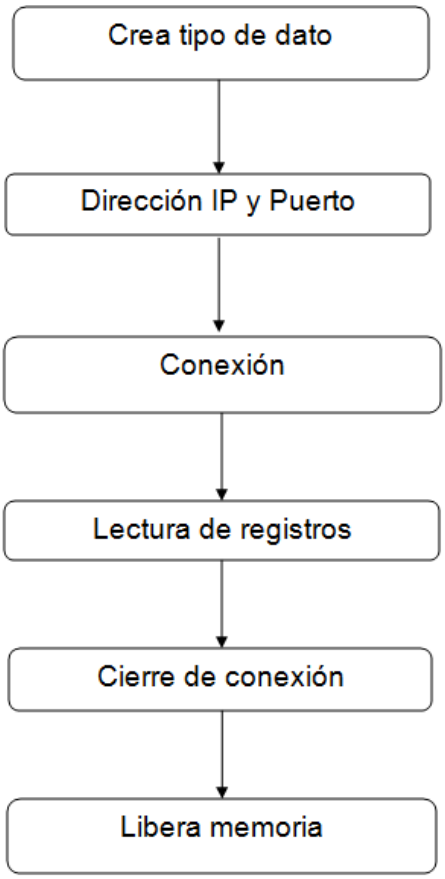


Figura 8 Diagrama de conexión Modbus.

El primer bloque de la Figura 8 crea el tipo de dato a utilizar, donde se alojará toda la información de conexión con el dispositivo a interrogar. En el segundo bloque se agrega la dirección IP y el puerto. El tercer bloque realiza la conexión con el servidor. El cuarto bloque realiza la lectura de los registros de memoria con la información solicitada. El quinto bloque cierra la conexión y el último bloque libera la memoria donde se guardó la información leída.

A continuación, una porción de código de ejemplo que representa lo descrito anteriormente.

```
modbus_t *mb;
uint16_t tab_reg[32];
mb = modbus_new_tcp("127.0.0.1", 1502);
modbus_connect(mb);

/* Read 5 registers from the address 0 */
modbus_read_registers(mb, 0, 5, tab_reg);

modbus_close(mb)
modbus_free(mb);
```

Figura 9 Ejemplo de una interrogación en general.

En la Figura 9 mb es un puntero a la estructura *modbus_t* donde se almacenará toda la información de conexión. La función *modbus_new_tcp()* declara la dirección IP del servidor y el puerto a utilizar. La función *modbus_connect()* establece la conexión hacia el servidor. La función *modbus_read_registers()* lee los registros de memoria del servidor desde la dirección 0 y continua leyendo los 5 registros consecutivos. Las lecturas son guardadas en el arreglo tab_reg. La función *modbus_close()* cierra la conexión y *modbus_free()* libera la memoria que se utilizó para guardar los registros leídos del servidor.

Los datos obtenidos en el ejemplo anterior son registros de 16 bits. Estos no pueden utilizarse de manera directa para generar un número flotante.

Para convertir los dos registros de 16 bits a un número flotante se usó la función *modbus_get_float()*. Esta función retorna un número flotante bajo el estándar IEEE 854-1987. La función debe apuntar a un arreglo de dos registros de 16 bits. Por ejemplo, si el primer registro tiene el valor 0x4465 y el segundo 0x229a el resultado leído seria 916.540649 [5].

La siguiente tabla describe los bloques de memoria leídos y su respectivo tamaño en bytes para cada tipo de medidor.

Cantidad de medidores	Tipo de medidor	variables leídas		Tamaño (bytes)
		Bloque	**Cantidad de registros**	
9	Shark 200-S	Bloque primario de lecturas	63	126
		Bloque primario de energia	72	144
		Bloque prmario de demanda	64	128
		Total	**199**	**398**
2	Shark 200	Bloque primario de lecturas	63	126
		Bloque primario de energia	72	144
		Bloque prmario de demanda	64	128
		Total	**199**	**398**
20	Shark 100-S	Bloque primario de lecturas	30	60
		Bloque primario de energia	18	36
		Bloque prmario de demanda	20	40
		Total	**68**	**136**

Tabla 2. Bloques de memoria interrogados.

2.1 Interrogación al bloque primario de lecturas.

De todo el mapa de memoria de los medidores se lee primeramente el bloque primario de lecturas. Los detalles de las variables extraídas se muestran en la Tabla 3, en la Tabla 4 y en la Tabla 5. La Tabla 3 y la Tabla 4 corresponden a los segmentos de memoria Modbus de los medidores Shark 200-S y Shark 200, respectivamente. Entre otros, la Tabla 3 y la Tabla 4 contienen los valores los valores de voltaje, corriente, y potencia activa. La Tabla 5 contiene los segmentos de memoria Modbus de los medidores Shark 100-S.

Cantidad de medidores	Tipo de medidor	variables leídas		
		registro Nº	Dirección	
9	Shark 200-S	1	999	Voltaje A-N
		2	1001	Voltaje B-N
		3	1003	Voltaje C-N
		4	1005	Voltaje A-B
		5	1007	Voltaje B-C
		6	1009	Voltaje C-A
		7	1011	Corriente A
		8	1013	Corriente B
		9	1015	Corriente C
		10	1017	Potencia Activa Trifasica Total
		11	1019	Potencia Reactiva Trifasica Total
		12	1021	Potencia Aparente Trifasica Total
		13	1023	Factor de Potencia Trifasico Total
		14	1025	Frecuencia
		15	1027	Corriente Neutro
		16	1029	Potencia Activa Fase A
		17	1031	Potencia Activa Fase B
		18	1033	Potencia Activa Fase C
		19	1035	Potencia Reactiva Fase A
		20	1037	Potencia Reactiva Fase B
		21	1039	Potencia Reactiva Fase C
		22	1041	Potencia Aparente Fase A
		23	1043	Potencia Aparente Fase B
		24	1045	Potencia Aparente Fase C
		25	1047	Factor de Potencia Fase A
		26	1049	Factor de Potencia Fase B
		27	1051	Factor de Potencia Fase C

Tabla 3. Variables leídas, bloque primario de lecturas, medidores Shark 200-S

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Direccón	
2	Shark 200	1	999	Voltaje A-N
		2	1001	Voltaje B-N
		3	1003	Voltaje C-N
		4	1005	Voltaje A-B
		5	1007	Voltaje B-C
		6	1009	Voltaje C-A
		7	1011	Corriente A
		8	1013	Corriente B
		9	1015	Corriente C
		10	1017	Potencia Activa Trifasica Total
		11	1019	Potencia Reactiva Trifasica Total
		12	1021	Potencia Aparente Trifasica Total
		13	1023	Factor de Potencia Trifasico Total
		14	1025	Frecuencia
		15	1027	Corriente Neutro
		16	1029	Potencia Activa Fase A
		17	1031	Potencia Activa Fase B
		18	1033	Potencia Activa Fase C
		19	1035	Potencia Reactiva Fase A
		20	1037	Potencia Reactiva Fase B
		21	1039	Potencia Reactiva Fase C
		22	1041	Potencia Aparente Fase A
		23	1043	Potencia Aparente Fase B
		24	1045	Potencia Aparente Fase C
		25	1047	Factor de Potencia Fase A
		26	1049	Factor de Potencia Fase B
		27	1051	Factor de Potencia Fase C

Tabla 4. Variables leídas, bloque primario de lecturas, medidores Shark 200

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
20	Shark 100-S	1	999	Voltaje A-N
		2	1001	Voltaje B-N
		3	1003	Voltaje C-N
		4	1005	Voltaje A-B
		5	1007	Voltaje B-C
		6	1009	Voltaje C-A
		7	1011	Corriente A
		8	1013	Corriente B
		9	1015	Corriente C
		10	1017	Potencia Activa Trifasica Total
		11	1019	Potencia Reactiva Trifasica Total
		12	1021	Potencia Aparente Trifasica Total
		13	1023	Factor de Potencia Trifasico Total
		14	1025	Frecuencia
		15	1027	Corriente Neutro

Tabla 5. Variables leídas, bloque primario de lecturas, medidores Shark 100

Para la interrogación a los registros de este bloque, se comienza en un registro en particular, para luego seguir con la lectura consecutiva de sus registros superiores como se muestra en la Figura 10. En este caso, el registro inicial a leer es el 999. Este contiene el valor de voltaje A-N como se muestra en las Tablas 3, 4, y 5.

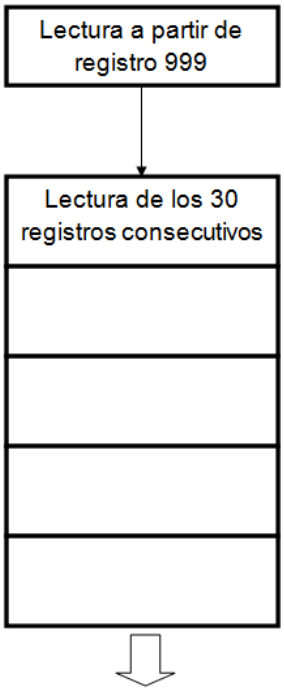


Figura 10. Diagrama de interrogación de registros.

A continuación se muestra un segmento de código de interrogación al bloque primario de lecturas de un medidor Shark 100-S.

```
rc = modbus_read_registers(ctx, 999, 30, tab_reg);
if (rc == -1) {
   fprintf(stderr, "Intento %s Fallo lectura Bloque Primario de Agronomia: %s | %s", argv[1],
modbus_strerror(errno), ctime(&horalocal));
   return -1;
}
```

Figura 11 Código de interrogación al bloque primario de lecturas.

Como se muestra en la Figura 11 la función modbus_read_registers comienza a leer desde el registro 999 y continua leyendo 30 registros consecutivos, para un total de 15 variables.

De manera similar para los medidores Shark 200-S y Shark 200, se leen 54 registros consecutivos, para un total de 27 variables leídas del bloque primario, como se muestra en las Tablas 3 y 4.

2.2 Interrogación al bloque primario de energía.

Las variables leídas del bloque primario de energía se muestran en las Tablas 6, 7 y 8 para los medidores Shark 200-S, Shark-200 y Shark 100-S, respectivamente.

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
9	Shark 200-S	1	1499	KW-h recibidos
		2	1501	KW-h entregados
		3	1503	KW-h Neto
		4	1505	KW-h Total
		5	1507	KVAR-h Positivo
		6	1509	KVAR-h Negativo
		7	1511	KVAR-h Neto
		8	1513	KVAR-h Total
		9	1515	KVA-h Total
		10	1517	KW-h recibidos Fase A
		11	1519	KW-h recibidos Fase B
		12	1521	KW-h recibidos Fase C
		13	1523	KW-h entregados Fase A
		14	1525	KW-h entregados Fase B
		15	1527	KW-h entregados Fase C
		16	1529	KW-h Neto Fase A
		17	1531	KW-h Neto Fase B
		18	1533	KW-h Neto Fase C
		19	1535	KW-h Total Fase A
		20	1537	KW-h Total Fase B
		21	1539	KW-h Total Fase C
		22	1541	KVAR-h Positivo Fase A
		23	1543	KVAR-h Positivo Fase B
		24	1545	KVAR-h Positivo Fase C
		25	1547	KVAR-h Negativo Fase A
		26	1549	KVAR-h Negativo Fase B
		27	1551	KVAR-h Negativo Fase C
		28	1553	KVAR-h Neto Fase A
		29	1555	KVAR-h Neto Fase B
		30	1557	KVAR-h Neto Fase C
		31	1559	KVAR-h Total Fase A
		32	1561	KVAR-h Total Fase B
		33	1563	KVAR-h Total Fase C
		34	1565	KVA-h Fase A
		35	1567	KVA-h Fase B
		36	1569	KVA-h Fase C

Tabla 6 Variables leídas, bloque primario de energía, medidores Shark 200-S

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
2	Shark 200	1	1499	KW-h recibidos
		2	1501	KW-h entregados
		3	1503	KW-h Neto
		4	1505	KW-h Total
		5	1507	KVAR-h Positivo
		6	1509	KVAR-h Negativo
		7	1511	KVAR-h Neto
		8	1513	KVAR-h Total
		9	1515	KVA-h Total
		10	1517	KW-h recibidos Fase A
		11	1519	KW-h recibidos Fase B
		12	1521	KW-h recibidos Fase C
		13	1523	KW-h entregados Fase A
		14	1525	KW-h entregados Fase B
		15	1527	KW-h entregados Fase C
		16	1529	KW-h Neto Fase A
		17	1531	KW-h Neto Fase B
		18	1533	KW-h Neto Fase C
		19	1535	KW-h Total Fase A
		20	1537	KW-h Total Fase B
		21	1539	KW-h Total Fase C
		22	1541	KVAR-h Positivo Fase A
		23	1543	KVAR-h Positivo Fase B
		24	1545	KVAR-h Positivo Fase C
		25	1547	KVAR-h Negativo Fase A
		26	1549	KVAR-h Negativo Fase B
		27	1551	KVAR-h Negativo Fase C
		28	1553	KVAR-h Neto Fase A
		29	1555	KVAR-h Neto Fase B
		30	1557	KVAR-h Neto Fase C
		31	1559	KVAR-h Total Fase A
		32	1561	KVAR-h Total Fase B
		33	1563	KVAR-h Total Fase C
		34	1565	KVA-h Fase A
		35	1567	KVA-h Fase B
		36	1569	KVA-h Fase C

Tabla 7 Variables leídas, bloque primario de energía, medidores Shark 200

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
20	Shark 100-S	1	1099	KW-h recibidos
		2	1101	KW-h entregados
		3	1103	KW-h Neto
		4	1105	KW-h Total
		5	1107	KVAR-h Positivo
		6	1109	KVAR-h Negativo
		7	1111	KVAR-h Neto
		8	1113	KVAR-h Total
		9	1115	KVA-h Total

Tabla 8 Variables leídas, bloque primario de energía, medidores Shark 100-S

De las tablas anteriores se observa que para los medidores Shark 200-S y Shark 200 se leen 36 variables de 2 registros cada una, con un total de 72 registros. Para los medidores Shark 100-S se leen 9 variables de 2 registros para un total de 18 registros.

2.3 Interrogación al bloque primario de demanda

El bloque primario de demanda no había sido implementado en [1]. Este bloque es importante para determinar el cargo por demanda dentro de la facturación. Las variables leídas del bloque primario de demanda se muestran en las tablas 9, 10 y 11.

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
9	Shark 200-S	1	2005	Watts Positivos, Trifasico, Promedio
		2	2007	VARs Positivos, Trifasico, Promedio
		3	2009	Watts Negativo, Trifasico, Promedio
		4	2011	VARs Negativos, Trifasico, Promedio
		5	2013	VA, Trifasico, Promedio
		6	2015	Factor de Potencia Positivo, Trifasico, Promedio
		7	2017	Factor de Potencia Negativo, Trifasico, Promedio
		8	2019	Corriente Neutro, Promedio

Tabla 9 Variables leídas, bloque primario de demanda, medidores Shark 200-S

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
2	Shark 200	1	2005	Watts Positivos, Trifasico, Promedio
		2	2007	VARs Positivos, Trifasico, Promedio
		3	2009	Watts Negativo, Trifasico, Promedio
		4	2011	VARs Negativos, Trifasico, Promedio
		5	2013	VA, Trifasico, Promedio
		6	2015	Factor de Potencia Positivo, Trifasico, Promedio
		7	2017	Factor de Potencia Negativo, Trifasico, Promedio
		8	2019	Corriente Neutro, Promedio

Tabla 10 Variables leídas, bloque primario de demanda, medidores Shark 200

Cantidad de medidores	Tipo de medidor	variables leídas		
		Registro Nº	Dirección	
20	Shark 100-S	1	2005	Watts Positivos, Trifasico, Promedio
		2	2007	VARs Positivos, Trifasico, Promedio
		3	2009	Watts Negativo, Trifasico, Promedio
		4	2011	VARs Negativos, Trifasico, Promedio
		5	2013	VA, Trifasico, Promedio
		6	2015	Factor de Potencia Positivo, Trifasico, Promedio
		7	2017	Factor de Potencia Negativo, Trifasico, Promedio

Tabla 11 Variables leídas, bloque primario de demanda, medidores Shark 100

A diferencia de la lectura en el bloque primario de energía, en el bloque primario de demanda se decidió leer únicamente las variables de valores trifásico promedio. Para efectos del cálculo de demanda máxima las variables trifásicas promedio son las únicas necesarias.

2.4 Interrogación *Broadcast*

Como se mostró en la Figura 2, en un principio, la interrogación se realizaba de forma secuencial, un medidor a la vez. Sin preocuparse de si la interrogación era o no exitosa. Sin embargo, los enlaces de radio mediante WiFi mostraron poca robustez. Conduciendo a la pérdida de datos. Como se detalló en la sección de antecedentes, se identificaron dos tipos de errores, todos atribuibles a la variación en la calidad de los enlaces de radio. El primer error identificado no permitía el establecimiento de una conexión TCP, como se mostró en la Figura 6. El segundo error si permitía el establecimiento de una conexión TCP, pero nunca se llegaba a finalizar con satisfacción la transferencia de los datos debido a que no existían ninguna persistencia por parte del protocolo Modbus, como se muestra en la Figura 7(b).

Como solución a los problemas antes descritos se creó un *daemon*. Si por alguna razón hay un fallo en el establecimiento de una conexión TCP, el *daemon* persiste tantas veces como se haya programado, buscando el establecer la conexión cliente-servidor. Por otra parte, para superar el problema de pérdida de transferencia de datos, el *daemon* interroga a los medidores de una manera diferente, todos al mismo tiempo. Esta idea se denominó interrogación tipo broadcast. Y permitió tener control sobre el tiempo de persistencia en la interrogación de cada medidor.

La Figura 12 muestra cómo se ejecuta de manera simultánea la interrogación hacia la red. Este nuevo mecanismo de interrogación permite definir tiempos de persistencia en la interrogación de cada medidor.

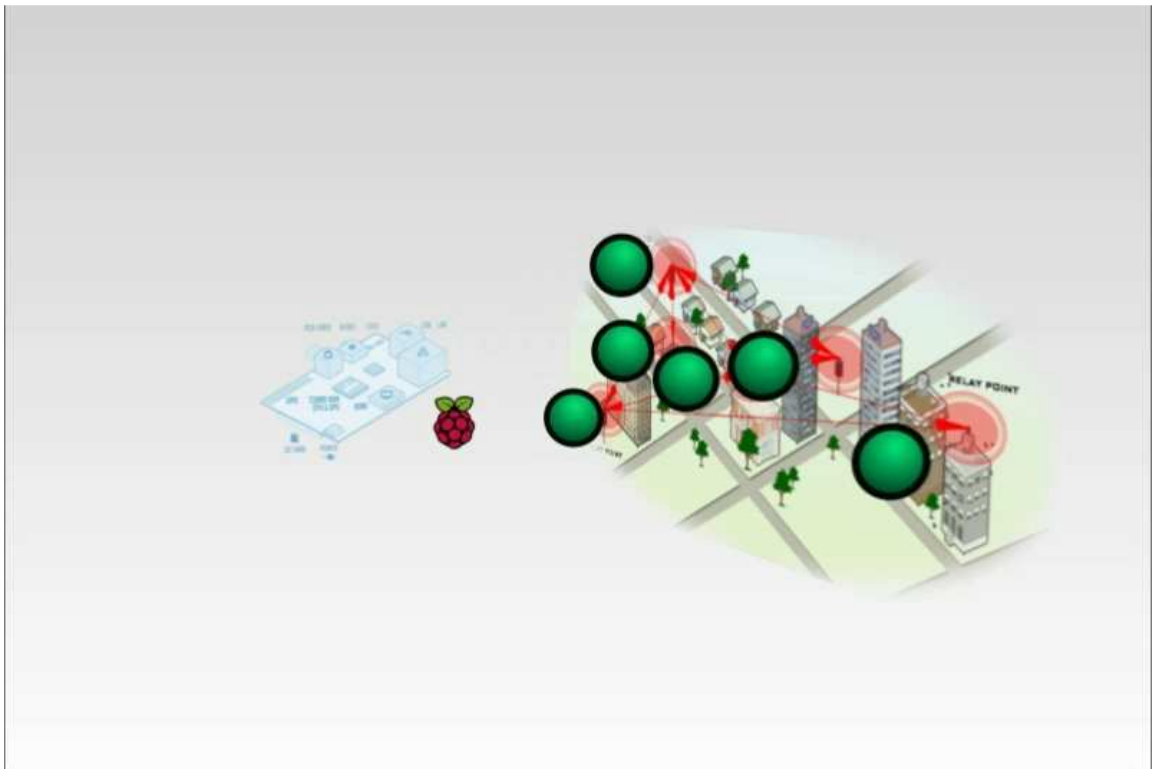


Figura 12 Interrogación simultánea a cada medidor.

Los periodos de interrogación se realizan cada 15 minutos debido a que se pretende enviar las lecturas a una aplicación en la nube, pero la red de medidores puede ser interrogada en intervalos de hasta 1 segundo.

La Figura 13 muestra en un flujograma el funcionamiento del *daemon*, con la interrogación para cada medidor.

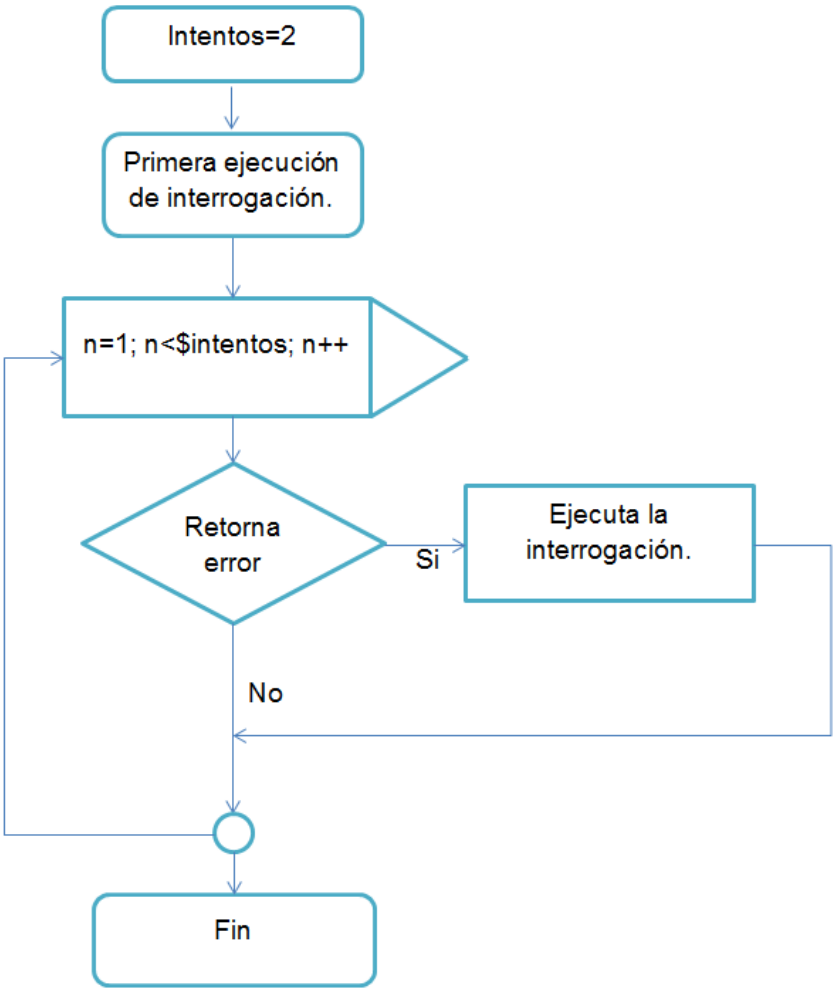


Figura 13 Flujograma *daemon*.

A continuación la porción de código del *daemon* de interrogación.

```
#!/bin/bash

intentos=2
{
(./agronomia 1 )
for ((n=1; n<$intentos; n++))
 do
   if [ $? -ne 0 ]
   then
    (./agronomia $((n+1)))
   fi
 done
}&
```

Figura 14 Script de interrogación para medidor Agronomía.

La Figura 14 ejecuta el programa de interrogación para el medidor ubicado en la facultad de agronomía. Se envía como el argumento el número 1, para identificarlo como el primer intento en el archivo log generado en caso de fallo.
$? obtiene el valor de retorno (*return* de la función *main*) devuelto por el programa ./agronomía. Si el programa retorna un valor diferente de cero significa que hubo un error y la tarea de interrogación no se completó. Por tanto se repite la interrogación hacia el medidor una vez más debido a que la variable intentos se definió con el valor de 2. De esta manera se logra reducir la pérdida de datos desde los medidores. Cada consulta se ejecuta en segundo plano dentro del bloque { }&.

2.5 Persistencia Modbus

Tal como se mostró en la Figura 7(b) existen situaciones donde se establece la conexión con el medidor pero el envío de datos falla. Para resolver este problema se utilizó una de las funciones de la librería Libmodbus. Ésta cuenta con una función que permite establecer el intervalo de tiempo usado para esperar por una respuesta. Si el tiempo de espera antes de recibir una respuesta se agota, se produce un error.

En la Figura 15 se muestra como la función *modbus_set_response_timeout()* espera a que el envío del paquete se realice con éxito. El circulo amarrillo representa la función de persistencia en funcionamiento. El círculo verde representa las interrogaciones que se realizan con éxito. Mientras que la interrogación a los demás medidores termina satisfactoriamente de acuerdo a la calidad de su enlace, la función de persistencia puede permanecer esperando reconexión de acuerdo a su programación.

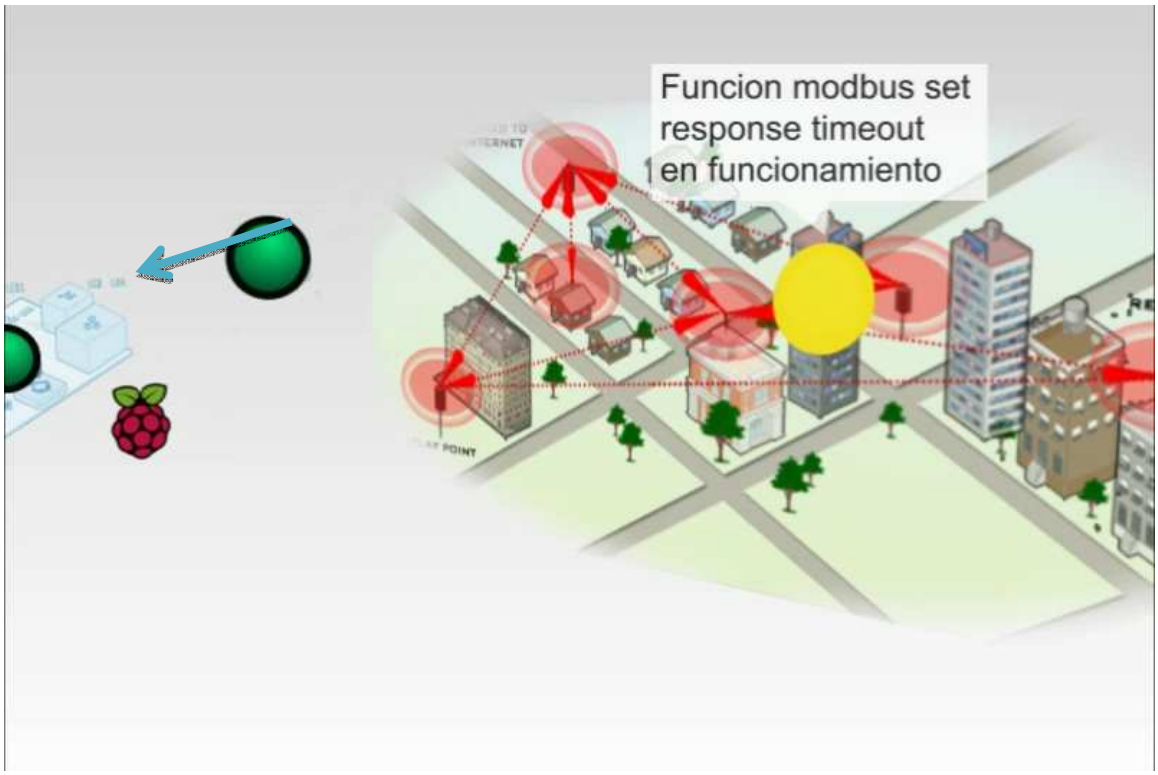


Figura 15. Representación de la función de persistencia de enlace.

A continuación, la porción de código que representa la ilustración anterior.

```
struct timeval response_timeout;
response_timeout.tv_sec = 120;
response_timeout.tv_usec = 0;
modbus_set_response_timeout(ctx, &response_timeout);
```

Figura 16. Programación de tiempo de espera.

En la Figura 16 se muestra la estructura *response_timeout* que define un tiempo de espera de 120 segundos. Esto reduce en gran medida la pérdida de datos. Como se mostró en la Figura 2 no existía ninguna persistencia, en cambio en esta se dispone de 120 segundos para esperar por la respuesta de recibido.

La función *modbus_set_response_timeout()* solamente entra en funcionamiento cuando la conexión con la capa de aplicación Modbus ha sido establecida, es decir, se efectuó con éxito el establecimiento de conexión de tres pasos y se logró conectar con el medidor.

Una vez que los datos de la interrogación se encuentran en memoria, deben de ser guardados de manera definitiva en un sistema de base de datos. Este sistema de base de datos permite que la información sea enviada de una manera

programada hacia una aplicación web para su visualización. El desarrollo de la aplicación web tiene dos ventajas. Tener la base de datos local respaldada en la Internet y la visualización de los datos de manera gráfica por cualquier persona con acceso a la Internet.

Capítulo 3. Base de datos y aplicaciones *cloud computing*

3.1 Base de datos MySQL

En [1], el hardware utilizado para la interrogación a los medidores y sistema de base de datos fue una PC marca Dell modelo Vostro 3000. El uso de este hardware representaba un exceso en la capacidad de procesamiento, consumo de energía y espacio físico necesario. Como solución a esto se decidió utilizar la microcomputadora Raspberry Pi Versión B.

Raspberry Pi es una microcomputadora de hardware libre. Ésta cuenta con un microprocesador ARM11 a 700Mhz, 512 MB de memoria RAM, un puerto para tarjeta SD de hasta 32GB, puerto Ethernet de 100Mbits entre otras características. Este hardware corre en un sistema operativo Linux. Una de las ventajas de utilizar esta computadora es su bajo precio, consumo de tan solo 3.5 Watts y la capacidad de ser ubicada remotamente en una caja IP, NEMA o IEC.

Adicionalmente a la microcomputadora se le ha instalado un circuito electrónico reloj de tiempo real (*Real Time Clock*). Por defecto Raspberry Pi establece la hora únicamente cuando existe una conexión a Internet. Para la aplicación desarrollada es vital tener una hora sincronizada bajo cualquier circunstancia.

Adicionalmente al cambio del hardware se realizaron modificaciones en las tablas de base de datos diseñadas en [1]. Se agregaron los campos necesarios para almacenar los registros del bloque primario de demanda y se eliminaron los campos reservados para el bloque de distorsión armónica para los medidores que de momento no lo generan.

Se instaló el gestor de base de datos MySQL para el almacenamiento de datos junto con la API de MySQL para el lenguaje C.

Para guardar los datos hacia la base de datos se implementó la API dentro del código C como se muestra en la Figura 17.

```
MYSQL *conn;
MYSQL_RES *res;
MYSQL_ROW row;
char *server = "localhost";
char *user = "root";
char *password = "root";
char *database = "meters";
conn = mysql_init(NULL);

/* Connect to database */
 if (!mysql_real_connect(conn, server,
      user, password, database, 0, NULL, 0)) {
   fprintf(stderr, "%s\n", mysql_error(conn));
   exit(1);
 }
```

Figura 17 Conexión con base de datos desde codigo en C.

El código de la Figura 17 establece la conexión a la base de datos.

Una vez se han leído todas las variables, se copian a un string para guardarse en MySQL como se muestra en la Figura 18.

```
sprintf(query,"insert into Artes(Va,Vb,Vc) values ('%f','%f','%f') ",Voltaje_A, Voltaje_B, Voltaje_C);
```

Figura 18 Copia de variables con datos a la variable query.

El código de la Figura 18 copia al string query las variables Voltaje_A, Voltaje_B, Voltaje_C, a la tabla Artes de MySQL.

```
mysql_query(conn, query);
mysql_free_result(res);
mysql_close(conn);
```

Figura 19 Envío de datos, liberación de memoria y cierre de base de datos.

Las tres líneas de la Figura 19 sirven para enviar la variable query a MySQL, liberar la memoria y cerrar la conexión de la base de datos.

Una de las innovaciones introducidas en este trabajo de graduación consistió en enviar a una aplicación en la web para que puedan ser visibles a cualquier persona. Además, la base de datos en la web sirve como una copia de seguridad de la base de datos local. Esta aplicación tendría un gran impacto social a la comunidad universitaria ya que los datos pueden ser monitorizados por cualquier persona haciendo conciencia sobre el gasto de la energía que se consume mes a mes.

Para enviar todos los datos generados de manera continua por los medidores, la mejor solución es hacer uso de la computación en la nube. A diferencia de un *web hosting,* ésta permite manejar grandes cantidades de base de datos en una aplicación web.

Para desarrollar la aplicación web se decidió utilizar la plataforma como un servicio de Google: *Google App Engine*. Esta ofrece una cuota gratuita diaria para lectura y escritura. Además ofrece complementos como *memcache, email, task queues, local static file storage* de manera gratuita.

3.2 Google Datastore

Los datos son almacenados en el sistema de base de datos de *Google* llamado, *Google Datastore*. *Datastore* forma parte de *Google App Engine* y cuenta con una tarifa de 50,000 operaciones de escritura diaria gratis y 50,000 operaciones de lectura diaria gratuita [6]. La cantidad de variables leídas durante el día se muestran en la Tabla 12 donde se puede observar que no es posible guardar todas las variables en la aplicación web utilizando únicamente la cuota gratuita por día.

Cantidad	Medidores	Variables	Interrogaciones por hora	Total por dia
9	Shark 200-S	36	4	31104
2	Shark 200	36	4	6912
20	Shark 100	15	4	28800
			Total:	**66816**

Tabla 12 Variables generadas por día.

El total de variables guardadas (66,816) es mayor al número de operaciones de escritura gratuita por día (50,000). Es por ello que se decidió enviar únicamente las variables más importantes para efectos de facturación.

Ya que se utiliza únicamente la cuota gratuita proporcionada por *Google*, no se envían todas las variables de la base de datos a la web. Solamente se envían las variables de mayor relevancia para un usuario en general. Las variables enviadas se muestran en la Tabla 13.

variables enviadas a la aplicación web
Fecha_hora
Voltaje A-N
Voltaje B-N
Voltaje C-N
Corriente A
Corriente B
Corriente C
Potencia Activa Trifasica Total
KW-h Totales
Demanda máxima
Factor de Potencia Trifasico Total
Corriente Neutro

Tabla 13 Variables enviadas a aplicación web, igual para todos los modelos.

En la Tabla 14 se muestra el total de variables enviadas a la aplicación web.

Cantidad medidores	Variables	Interrogaciones por hora	Total por día
31	12	4	1488

Tabla 14 Variables enviadas a la aplicación web.

En la Tabla 14 se puede observar que la cantidad de variable a enviar se reduce a 1488, lo cual no excede la tarifa gratuita diaria proporcionada por *Google*.

Datastore es parte de *Google App Engine* y su utilización se basa en el lenguaje de programación *Python.*

El sistema de base de datos *Datastore* funciona de manera diferente a los sistemas SQL. *Datastore* contiene objetos de datos conocidos como entidades. Éstas se declaran de igual forma que una clase de *Python*. Una entidad es análoga a una tabla en SQL. Una entidad tiene una o más propiedades, valores con nombre de uno de los varios tipos de datos soportados: por ejemplo, una propiedad puede ser una cadena, un entero o una referencia a otra entidad.

Se creó una entidad para cada medidor. Esta es un objeto en App Engine. Y las variables que contiene la entidad son las propiedades del objeto. La Figura 20 muestra un esquema de la entidad de Agronomía en este sistema de base de datos.

Figura 20 Esquema de la entidad de Agronomía.

La Figura 21 muestra el código que crea la entidad de Agronomía para la aplicación web.

```
class Agronomia(db.Model):
   fecha_hora=db.DateTimeProperty()
   va=db.FloatProperty()
   vb=db.FloatProperty()
   vc=db.FloatProperty()
   ia=db.FloatProperty()
   ib=db.FloatProperty()
   ic=db.FloatProperty()
   FP=db.FloatProperty()
   Ineutro=db.FloatProperty()
   potencia=db.FloatProperty()
   energia=db.FloatProperty()
   demanda=db.FloatProperty()
```

Figura 21 Código de la entidad de Agronomía.

3.3 Desarrollo Web

Google App Engine puede ser usado por cualquier persona con una suscripción gratuita. Para registrar una aplicación web en éste servicio, únicamente se necesita tener una cuenta de correo electrónico válida. La dirección del servicio es: https://appengine.google.com/. Las aplicaciones de Gogle App Engine pueden escribirse en Python 2.7 pero también pueden ser desarrolladas en los lenguajes de programación Java, Go y PHP. Para desarrollar una aplicación utilizando *Python* se utiliza el *web application framework* de Google llamado *webapp2*. Éste *framework* ya viene instalado por defecto en el *Server Development Kit* de *App Engine.* Como se mencionó anteriormente, se utiliza el sistema de base de datos *Datastore* como sistema único de almacenamiento de datos. Para la interfaz de la

aplicación se utilizó el sistema de plantillas para Python Jinja2. Para visualizar la información con gráficos dinámicos se utilizó la API de *JavaScript, Google Chart Tools*. *Google Charts* provee una excelente forma de visualizar datos en la aplicación web. Una de las ventajas es que los gráficos son generados usando la tecnología HTML5/SGV, esto permite tener la visualización entre diferentes navegadores (incluyendo VML para versiones más antiguas de Internet Explorer) y visualización multiplataforma para compatibilidad con iPhone, iPad y Android [7]. Para poder seleccionar el periodo de interrogación y presentar múltiples graficas mediante tabs se utilizó el framework de JavaScript jQuery UI.

Para comenzar con el desarrollo se descargó el *Google App Engine SDK* para *Python* [8], que incluye una aplicación del servidor web que simula el entorno de App Engine, y las herramientas para implementar la aplicación en el entorno de producción de App Engine.

3.3.1 Creación de la aplicación

Se creó un directorio con el nombre de *tesisduque*. Todos los archivos de la aplicación residen en este directorio. La carpeta debe de ubicarse dentro del *SDK* de *Google App Engine.* Dentro de la carpeta tesisduque se creó el archivo de Python con el nombre *tesisduque.py* El contenido de este archivo se divide en cinco categorías: Preparación del programa, Manejador, Consulta, Registro y Home.

3.3.1.1 Preparación del programa

El archivo principal contiene la declaración de las librerías a utilizar y el enlace de la aplicación de Python con el *framework jinja2* para enviar contenido al navegador web como se muestra en la Figura 22.

```
import os
import re
import random
import hashlib
import hmac
from string import letters
import datetime
import gviz_api
from collections import namedtuple

import webapp2
import jinja2
```

```
from google.appengine.ext import db
from google.appengine.ext.db import polymodel

template_dir = os.path.join(os.path.dirname(__file__), 'templates')
jinja_env = jinja2.Environment(loader = jinja2.FileSystemLoader(template_dir),
                        autoescape = True)

app = webapp2.WSGIApplication([('/', Home),
                        ('/consulta', Consulta),
                        ('/registro', Registro),
                        ('/hecho', Hecho)
                        ],
                        debug=True)
```

Figura 22 Cabecera del archivo tesisduque.py.

3.3.1.2 La clase Manejador

La clase manejador contiene los métodos que interpretan los archivos html y escriben en la página web, como se muestra en la Figura 23

```
class Handler(webapp2.RequestHandler):
   def write(self, *a, **kw):
      self.response.out.write(*a, **kw)

   def render_str(self, template, **params):
      t = jinja_env.get_template(template)
      return t.render(params)

   def render(self, template, **kw):
      self.write(self.render_str(template, **kw))
```

Figura 23 Clase manejador.

El método *write* escribe en la página web de html. El método *render_str* interpreta la página web. El método *render* interpreta la página web y pasa como argumentos la información necesaria a la página.

3.3.1.3 La clase Consulta

La clase consulta permite la interrogación por parte del programa de sincronización de datos. Al interrogarse esta clase devuelve la fecha de la última fila contenida en la entidad solicitada. A partir de la fecha devuelta, el programa de sincronización puede cargar los datos nuevos a la aplicación. La Figura 24 muestra una porción de código de la clase Consulta.

```
class Consulta(Handler):
   def post(self):
      nombre=self.request.get('nombre')
       self.Db_entity=nombre()
      fecha = self.Db_entity.all().order('-fecha_hora').get()
      self.render('registro.html',   fecha=fecha.fecha_hora)
```

Figura 24 Clase Consulta.

3.3.1.4 La clase Registro

La clase registro recibe los datos de actualización por parte de la base de datos local y los ingresa en la entidad correspondiente de la aplicación web. La Figura 25 muestra este proceso.

```
class Registro(BlogHandler):

   def post(self):
      nombre=self.request.get('nombre')
      fecha_hora=datetime.datetime.strptime(
         self.request.get('fecha_hora'),"%Y-%m-%d %H:%M:%S")
      va=float(self.request.get('va'))
      vb=float(self.request.get('vb'))
      vc=float(self.request.get('vc'))
      ia=float(self.request.get('ia'))
      ib=float(self.request.get('ib'))
      ic=float(self.request.get('ic'))
      FP=float(self.request.get('FP'))
      Ineutro=float(self.request.get('Ineutro'))
      potencia=float(self.request.get('potencia'))
      energia=float(self.request.get('energia'))
      demanda=float(self.request.get('demanda'))

       self.Db_entity=nombre()
      self.Db_entity.registro(fecha_hora,va,vb,vc,ia,ib,ic,FP,Ineutro,potencia,energia,demanda)
         medidor.put()
         self.redirect('/hecho')
      else:
         self.error(404)
         return
```

Figura 25 Clase Registro.

3.3.1.5 La clase Home

La Figura 26 muestra una poción de código de la clase Home.

```
class Home(Handler):

  def get(self):
    fecha_inicial_default=datetime.datetime.now()-datetime.timedelta(days=30)
    fecha_hoy=datetime.datetime.now().strftime("%Y-%m-%d")


    self.render('home.html',
      select1=self.html_select,
      select2=self.html_variable,
      tab=self.tab_position('va'),
      fecha_inicial=fecha_inicial_default.strftime("%Y-%m-%d"),
      fecha_final=fecha_hoy)

  def post(self):
    self.nombre=self.request.get('nombre')
    self.variable=self.request.get('variable')
    self.fecha_inicial=datetime.datetime.strptime(self.request.get('fecha_inicial'),"%Y-%m-%d")
    self.fecha_final=datetime.datetime.strptime( self.request.get('fecha_final'),"%Y-%m-%d")

    self.render('home.html',
      select1=self.html_select,
      select2=self.html_variable,
      periodo=periodo,
      fecha_inicial=self.fecha_inicial.strftime("%Y-%m-%d"),
      fecha_final=self.fecha_final.strftime("%Y-%m-%d"),
      **json)
```

Figura 26 Porción de código de la clase Home.

La clase Home es la primera que se ejecuta cuando se accede a la aplicación web. Esta llamada es del tipo GET. Se ejecuta el método get(), ahí está definido el intervalo de tiempo a graficar por defecto y carga la página home.html. Cuando se envía una solicitud a graficar se utiliza el método POST. Éste solicita la información a la entidad correspondiente y envía la información devuelta hacia la página home.html para ser graficada.

3.3.2 Creación del archivo de configuración

Una aplicación de App Engine tiene un archivo de configuración llamado app.yaml. Entre otras cosas, este archivo describe cuales manejadores de scripts se deben

utilizar en cada dirección URL. El archivo de configuración se muestra en la Figura 27.

```
application: ues-energydashboard
version: 1
runtime: python27
api_version: 1

handlers:
- url: /static
  static_dir: static

- url: /.*
  script: tesisduque.app

libraries:
- name: jinja2
  version: latest

- name: PIL
  version: "1.1.7"

builtins:
- remote_api: on
```

Figura 27 Archivo de configuración app.yaml

En el archivo de configuración las partes que necesitan una mayor descripción son las siguientes: *application* es el nombre de la aplicación en la web. La opción *handlers* especifica la dirección de ficheros con contenido estático. *Script:* el nombre del archivo que contiene la aplicación. *Libraries* librerías adicionales utilizadas en la aplicación.

3.3.3 sincronización con la base de datos local

Para poder sincronizar la aplicación en la nube con la base de datos del servidor local se utiliza el módulo urllib para Python. Éste provee una interfaz de alto nivel para enviar datos a través de Internet [10]. La Figura 28 muestra cómo se realiza el envío de datos a la página web usando el módulo urllib.

```
params=urllib.urlencode({'nombre':nombre})
   f=urllib.urlopen("http://ues-energydashboard.appspot.com/consulta",params)
   ultimo_dato=f.read()
   print 'Ultimo dato en %s, %s' % (nombre, ultimo_dato)
   #cantidad de caracteres que contiene una fecha correcta: 19
```

```
if len(ultimo_dato)==19:
    contenedor=DB()
    array_data=contenedor.getData(nombre,ultimo_dato)
    for datos in array_data:
        print datos
        paquete=urllib.urlencode(datos)
        envio=urllib.urlopen("http:// ues-energydashboard.appspot.com/registro",paquete)
        if envio.read()!= 'Done':
            print "La fila con fecha %s no se subió al servidor" %(datos['fecha_hora'])
```

Figura 28 Sincronizador de base de datos.

La aplicación se ha creado de tal forma que permite que el programa de actualización le interrogue cual es la última fila de datos almacenada en la nube. A partir de esa respuesta, el método *getData* lee la base de datos local y envía la actualización a la aplicación web.

3.3.4 Visualización dinámica

Para la visualización de gráficos dinámicos se utilizó la API de JavaScript *Google Chart Tools. Chart Tools* cuenta con una gran cantidad de gráficos para cada necesidad. El tipo de grafico que se utilizó fue *Anotated Timeline [9]*. Ésta recibe la información desde la aplicación web y lo respresenta en la página de html. Una porción de codigo se muestra en la Figura 31.

```
<script type="text/javascript" src="//www.google.com/jsapi"></script>
<script type="text/javascript">
 function drawChartVA() {
  var voltaje_A = new
google.visualization.AnnotatedTimeLine(document.getElementById('visualizations-1'));
   var data1 = new google.visualization.DataTable({{ va|safe }}, 0.6);
   voltaje_A.draw(data1, {scaleType: 'maximized', 'thickness': 2, allValuesSuffix: ' Voltios'});
 }
 google.load('visualization', '1', {packages: ['annotatedtimeline'], callback: drawChartVA});
</script>
<div id="visualizations-1" style="width:400; height:250"></div>.
```

Figura 29 declaración de la función gráfica de Voltaje A.

En la primera línea de la Figura 32 se declara el uso de la API desde Internet. Se declara la función *drawChartVA()* para el grafico del voltaje fase A.

La variable voltaje_A realiza una instancia a la clase de visualización de nombre *google.visualization.AnnotatedTimeLine* Al mismo tiempo se declara el ID para ser llamado desde las etiquetas de html.

La variable *data1* almacena el arreglo de datos proveniente de la aplicación con la sintaxis *{{va|safe}}* a través del *framework* de *jinja2.* Este arreglo al mismo tiempo es transformado al tipo de arreglo *DataTable* utilizado específicamente por *Google Chart Tools* con la función *google.visualization.DataTable.*

La etiqueta <*div*> ubica la posición de la grafica dentro de la pagina. En este campo se debe especificar la altura y la anchura de la grafica de manera explícita.

3.3.5 Ejecución de la aplicación en el *SDK*

Para ejecutar la aplicación en el servidor de desarrollo local, la Figura 29 muestra la sintaxis utilizada.

```
/google_appengine/dev_appserver.py tesisduque/
```

Figura 30 Ejecución de la aplicación en el SDK.

3.3.6 Ejecución de la aplicación en la nube

Una vez probado en el servidor de desarrollo, se procede a subir a la web usando la sintaxis mostrada en la Figura 30.

```
/google_appengine/appcfg.py update tesisduque/
```

Figura 31 Envío de la aplicación a la nube.

La dirección de la versión final de la aplicación desarrollada es: http://ues-energydashboard.appspot.com La Figura 31 muestra una captura de la aplicación corriendo.

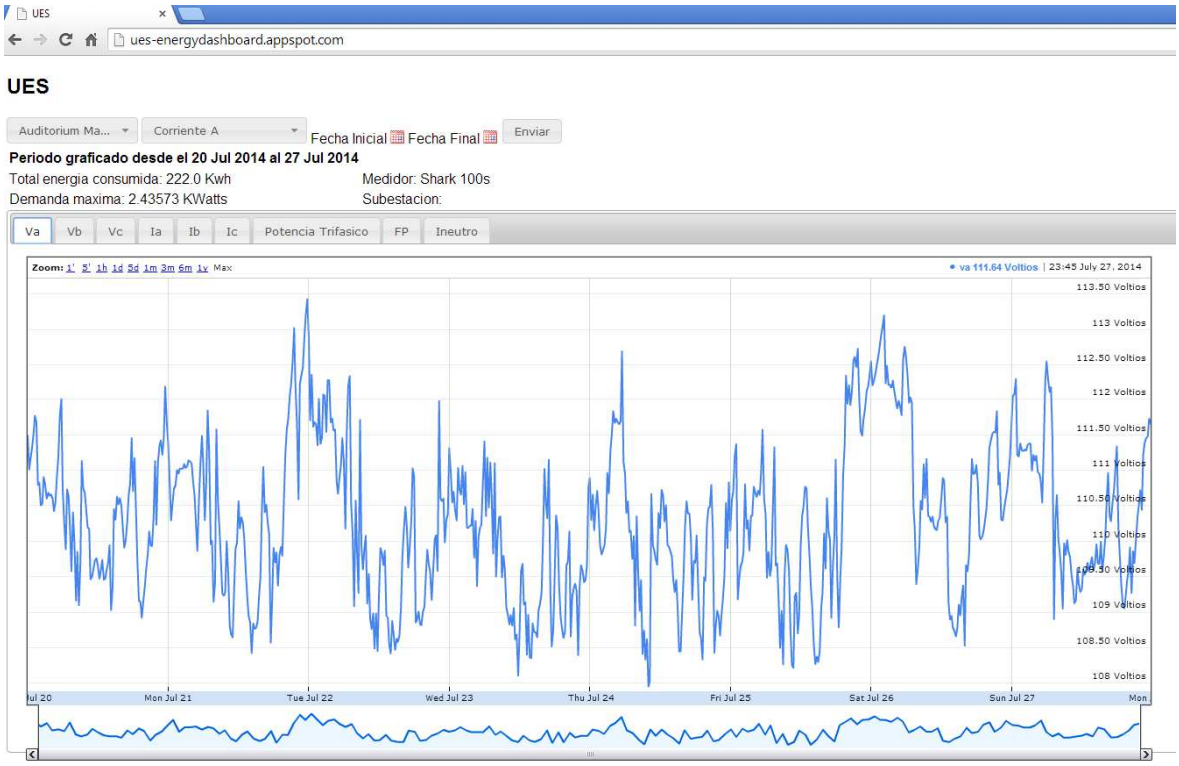


Figura 32 Aplicación web ues-energydashboard.appspot.com.

Como se mencionó anteriormente, la plataforma ofrece una cuota de 50,000 operaciones de lectura diarias de manera gratuita. Al hacer uso de la aplicación, es posible que se presente el error mostrado en la Figura 33. Esto significa que la tarifa de lectura para ese día se ha agotado. La aplicación puede volver a utilizarse el día siguiente.

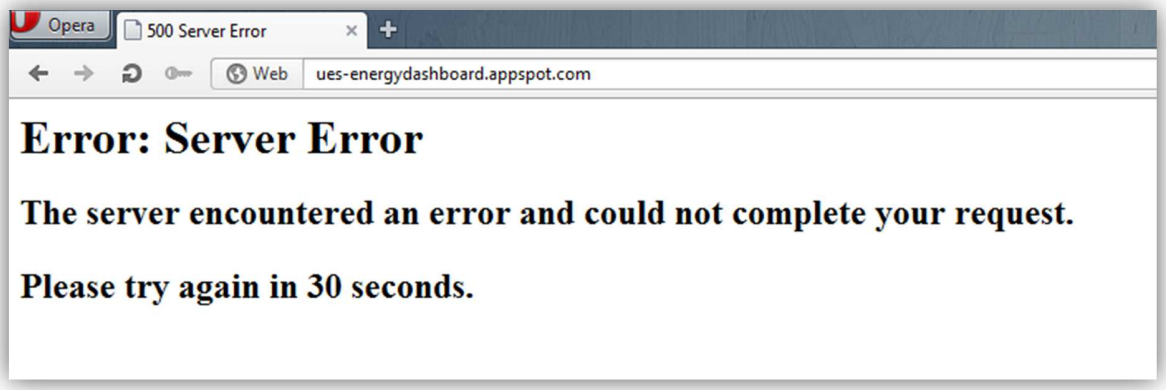


Figura 33 Uso de la aplicación agotado por ese día.

4. Conclusiones y líneas futuras

4.1 Conclusiones

- La optimización del software de comunicación, basado en el protocolo Modbus, de la red de medidores de energía eléctrica en el campus de la Universidad de El Salvador, permitió tener una mayor consistencia en la información almacenada en la base de datos. Ésta optimización permitió superar en gran medida la perdida de información en la interrogación a la red de medidores, a pesar de la debilidad en los enlaces.

- La instalación del sistema de interrogación y el sistema de base de datos MySQL en la microcomputadora Raspberry Pi genero excelentes resultados de rendimiento y estabilidad. Logrando así minimizar un consumo mayor de energía eléctrica por parte de un sistema de hardware de mayor capacidad.

- La utilización de una plataforma de *Google App Engine* permitió el desarrollo de una aplicación web completa y funcional sin la necesidad de adquirir o administrar el hardware del servidor. Siendo una opción factible para proyecto de desarrollo tecnológicos de bajo costo orientados a la web.

- La disponibilidad de una herramienta de monitoreo del consumo de energía eléctrica del campus de la Universidad de El Salvador, representa un gran aporte tecnológico para la comunidad universitaria. Con esto, ahora se puede acceder a la información del consumo de energía de manera pública a través de la Internet.

4.2 Líneas Futuras

- Realizar investigaciones sobre la estabilidad del sistema operativo ejecutándose en los routers inalámbricos MP01. En ocasiones la adquisición de los datos fallaba debido a que los routers dejaban de funcionar de manera momentánea.

- Seguir en el desarrollando de la aplicación web. La limitación en la cuota gratuita brindada por Google limita a la cantidad de veces que la aplicación web pueda ser interrogada por un cliente. *Google App Engine* ofrece diferentes herramientas de desarrollo para poder ahorrar el consumo de operaciones realizadas internamente.

- Incorporar más medidores de energía para las subestaciones no incluidas en la red. Medidores que cuenten con el protocolo de comunicación Modbus/TCP podrían incorporarse al sistema con facilidad.

- Implementar un script que sincronice eventualmente la RTC mediante un servidor NTP. Aunque una RTC es bastante confiable, con el tiempo puede perder la exactitud de la hora.

- Utilizar una micro SD Industrial Grade de alto rendimiento. Esto debido a que la micro SD está teniendo el papel principal del sistema de almacenamiento para la base de datos. La memoria SD utilizada actualmente no ha sido diseñada para operar en ambientes de lectura y escritura continua. Mientras tanto se recomienda realizar respaldos de la base de datos de manera periódica.

- Implementar la red de medidores dentro de la nueva infraestructura de red con la que cuenta la Universidad de El Salvador. Esto sería la evolución de la utilización de la red inalámbrica a la implementación dentro de la moderna red de fibra óptica de la Universidad de El Salvador.

ANEXOS

Anexo A. Comunicación Raspberry Pi y simulador de medidor de energía usando el protocolo Modbus TCP/IP

Para implementar una comunicación Modbus sobre TCP/IP se necesita de por lo menos dos dispositivos: un esclavo y un maestro para transferir la información. Esto podría realizarse usando dos PCs comunicándose entre sí, usando tarjetas estándar de Ethernet, o una PC comunicándose con un sensor o dispositivo con una microcomputadora.

Se realizó una prueba mediante una PC que establece la función de servidor corriendo un programa simulador de medidor Modbus. Éste represento al medidor de energía, el cual contiene los datos en registros de memoria.

Diagslave Modbus Slave Simulator es un pequeño programa de líneas de comando basado en un simulador de esclavo Modbus. Diagslave Modbus Slave Simulator es software libre y está disponible para Windows o Linux.

A.1 Ejecutando Diasgslave 2.12 en Linux

Primero ingresar a la página *http://www.modbusdriver.com/diagslave.html* y descargar el archivo ***diasgslave.2.12.zip*** (es posible que la versión disponible en ese momento sea diferente)

Para extraer el archivo se puede hacer desde el entorno gráfico o a través de la terminal con el comando

```
$ unzip diasgslave.2.12.zip
```

Si no se tiene instalado el programa zip puede instalarse con el siguiente comando

```
$ sudo apt-get install zip
$ sudo apt-get install unzip
```

Si es usuario Red Hat Linux/Fedora debe usarse los siguientes comandos

```
$ yum install zip
$ yum install unzip
```

Luego ingresar a la carpeta /**diagslave.2.12/linux** donde está el archivo binario y brindar los permisos de ejecución con el comando

```
$ chmod +x diagslave
```

Ejecutar el programa con los permisos de super usuario y comandos de ejecución con la siguiente sintaxis

```
$sudo ./diagslave -m tcp
```

Donde -m tcp especifica el contexto modbus a utilizar.

Cuando ya se está ejecutando el programa se mostrará un diálogo como el siguiente.

```
diagslave 2.12 - FieldTalk(tm) Modbus(R) Diagnostic Slave Simulator
Copyright (c) 2002-2012 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: MODBUS/TCP
Slave configuration: address = -1, master activity t/o = 3.00
TCP configuration: port = 502, connection t/o = 60.00

Server started up successfully.
Listening to network (Ctrl-C to stop)
.....
```

El programa está listo para recibir una petición de datos por parte del dispositivo maestro.

Anexo B. Descripción de código de interrogación en C y compilación usando la librería Libmodbus desde Raspberry Pi.

Para este ejemplo se utiliza una PC con el sistema operativo Linux Ubuntu 12.04. Pero el ejemplo es aplicable para cualquier sistema operativo Linux.

B.1 Código del programa.

A continuación se muestra un programa de ejemplo en lenguaje C que establece la función de cliente (master) realizando la petición de registros a un esclavo(Servidor) mediante el protocolo Modbus usando el contexto TCP/IPv4. Este cuenta con la descripción de cada una de las líneas de código.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <modbus.h>

int main(void)
{
modbus_t *ctx;       //Crea puntero
 uint16_t tab_reg[32]; //Crea un arreglo unsigned long int 16-bit para colocar los datos
 int rc,i;                //Contadores de registro

/* Establecer un contexto MODBUS */
 ctx = modbus_new_tcp("192.168.1.6", 502); //Usando contexto TCP, ip y puerto
if (ctx == NULL) {
   fprintf(stderr, "No pudo ubicar el contexto libmodbus\n");
   return -1;
  }

/* Establece la conexion MODBUS*/
modbus_connect(ctx);
if (modbus_connect(ctx) == -1) {
   fprintf(stderr, "La conexion fallo: %s\n", modbus_strerror(errno));
   modbus_free(ctx);
   return -1;
  }
/* Lee 10 registros desde la dirección 0 */
 rc=modbus_read_registers(ctx, 0, 10, tab_reg); //Lee registros
if (rc == -1) {
   fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
   return -1;
   }

for (i=0; i < rc; i++) {
   printf("reg[%d]=%d (0x%X)\n", i, tab_reg[i], tab_reg[i]);
```

```
}
/* Cierra la conexion MODBUS */
 modbus_close(ctx);
 modbus_free(ctx);
return(0);
}
```

En este programa se incluyen las librerías estándar de C junto con la librería modbus.h.

De acuerdo con el protocolo, un dispositivo Modbus solo acepta mensajes que contienen su identificativo Modbus o el número Broadcast específico. El ID del esclavo solamente es necesario en TCP si el mensaje tiene que localizar al dispositivo en una red.

modbus_new_tcp("192.168.1.6", 502) Crea el contexto Modbus para TCP/IPv4. Esta función debe asignar e inicializar la estructura *modbus_t* para comunicarse con el Servidor Modbus TCP/IPv4

La IP especifica la dirección del servidor hacia donde el cliente quiere establecer la conexión. Por motivos prácticos se ha establecido la IP 192.168.1.6.
Para usar el puerto por defecto (502) también puede usarse la etiqueta MODBUS_TCP_DEFAULT_PORT. Es conveniente usar puertos arriba de 1024 porque no necesita tener privilegios de administrador.

La función *modbus_new_tcp()* debe devolver un puntero a la estructura *modbus_t* si tuvo éxito, de lo contrario devuelve NULL y establece el error *EINVAL* La IP es inválida.

modbus_set_slave(ctx, SERVIDOR) Establece el número del Slave en el contexto TCP.
Cuando se requiere que todos los dispositivos en la red Modbus reciban la petición debe usarse la dirección Broadcast como MODBUS_BROADCAST_ADDRESS.

La función *modbus_connect(ctx)* establece la conexión con el Servidor Modbus a través de la red. La función debe de devolver 0 si tuvo éxito. De lo contrario devuelve -1 y establece el error

La funcion *modbus_read_registers(ctx, 0, 10, tab_reg)* lee el contenido de 10 registros comenzando desde la dirección 0 del dispositivo remoto. El resultado de la lectura es guardado en el arreglo *tab_reg* como palabras de 16 bit.

La función *modbus_read_registers()* devuelve el número de los registros leídos si tuvo éxito, de lo contrario devolverá -1 y establece el error. Un posible error puede ser *EMBMDATA* Demasiados registros solicitados.

modbus_close(ctx) cierra la conexión establecida con el servidor.
modbus_free(ctx) Libera la información alojada en la estructura *modbus_t*

B.2 Compilación del programa

Para poder compilar el programa se usa el compilador de C gcc con la siguiente sintaxis

```
$gcc lectura.c -o lectura `pkg-config --libs --cflags libmodbus`
```

Dónde lectura.c es el nombre del archivo que contiene el código. Lectura es el nombre que se le dará al ejecutable. `*pkg-config --libs --cflags libmodbus*` Es una herramienta de ayuda usada para compilar aplicaciones y librerías, ayuda a insertar las opciones de compilación correctamente en la línea de comando para que el programa pueda usarlas.

Si a la hora de compilar el programa se presenta el siguiente error

```
libmodbus.so.5 cannot open shared object file:No such file or directory
```

Debe de brindarse los privilegios de super usuario a la librería *ldconfig* con el siguiente comando.

```
$sudo ldconfig
```

Si la ejecución del programa no presento ningun problema, se ejecutará el archivo generado con el siguiente comando:

```
$ ./lectura
reg[0]=0 (0x0)
reg[1]=0 (0x0)
reg[2]=0 (0x0)
reg[3]=0 (0x0)
reg[4]=0 (0x0)
reg[5]=0 (0x0)
reg[6]=0 (0x0)
reg[7]=0 (0x0)
reg[8]=0 (0x0)
reg[9]=0 (0x0)
```

Los registros mostrados son las lecturas de datos realizadas al medidor, en este caso al emulador.

La información mostrada por parte del medidor es la siguiente

```
Server started up successfully.
Listening to network (Ctrl-C to stop)
........
validateMasterIpAddr: accepting connection from 192.168.1.5
Slave 1: readHoldingRegisters from 1, 10 references
```

Brinda la información de los registros solicitados. La IP 192.168.1.5 fue la definida al cliente que realiza la interrogación.

Anexo C. Instalación sistema operativo a Raspberry Pi

C.1 Acerca de la Raspberry Pi

Raspberry Pi es una micro computadora de bajo coste desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz, un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM aunque originalmente en su lanzamiento fue de 256 MB de memoria RAM. El diseño no incluye un disco duro o una unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación o carcasa.

La fundación Raspberry Pi da soporte para descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS y Arch Linux; y promueve principalmente el aprendizaje del lenguaje de programación Python y otros lenguajes como Tiny BASIC, C y Perl.

Raspbian es un sistema operativo basado en Debian 7 Wheezy el cual se ha convertido en el sistema operativo más utilizado para la Raspberry Pi, La distribución Debian 7 tiene soporte para microprocesadores ARM, por esa razón se adaptó con facilidad.

C.2 Instalación en una memoria SD

Existen diferentes métodos para instalar el sistema operativo en la memoria SD, en este caso se realizará la instalación usando Linux Ubuntu 12.04 bajo los siguientes pasos.

1. descargar la imagen del sistema operativo desde la página http://www.raspberrypi.org/downloads
Para este caso se descargó la versión 2013-02-09-wheezy-raspbian.zip

2. Se recomienda verificar que la descarga no presenta ningún error haciendo uso del Hash de seguridad brindado en la página de descarga. Asumiendo que el archivo zip fue descargado en la carpeta home (~/)

```
sha1sum ~/2013-02-09-wheezy-raspbian.zip
```

Esto mostrará un número hexadecimal el cual debe de coincidir con el SHA-1 brindado en la página de descarga

3. Extraer la imagen con:

```
unzip ~/2013-02-09-wheezy-raspbian.zip
```

4. Insertar la memoria SD en la computadora

5. Instalar el programa ImageWriter con el siguiente comando

```
~$ sudo apt-get install usb-imagewriter
```

6. Correr el programa con

```
~$ sudo imagewriter
```

7. Seleccionar la imagen, en este caso *2013-02-09-wheezy-raspbian.img* y el dispositivo de memoria SD

Una vez que la imagen ha sido instalada con éxito puede ser iniciada desde la Raspberry.

Anexo D. Instalación de la librería Libmodbus en sistema operativo Linux

D.1 Acerca de Libmodbus

Libmodbus es una librería de software libre para los sistemas operativos Linux, Mac OS X, FreeBSD, QNX y Win32 que envía y recibe datos conforme al protocolo Modbus. Esta librería está escrita en C y soporta comunicación RTU(Serial) y TCP (Ethernet).

D.2 Instalación de Libmodbus

Para la instalación de la librería Libmodbus, visite el sitio oficial http://libmodbus.org/ donde podrá descargar la última versión disponible.

El archivo descargado puede tener el nombre como libmodbus-3.0.x.tar.gz donde la x depende de la versión disponible al momento, esta librería puede ser instalada en cualquier distribución Linux, pero en este caso la instalación se realizará para la distribución Debian o Ubuntu.

Para la instalación primeramente se desempaqueta el archivo .tar.gz para lo cual necesita tener instalado el paquete de compilación llamado *built-essential*.

Para instalar este paquete de compilación, debe abrir un Terminal e ingresar el siguiente Comando:

```
sudo apt-get install build-essential
```

Aquí se ha de ingresar nuestra Contraseña de Root de Ubuntu.

Luego se procede a descomprimir el archivo empaquetado con el comando

```
tar -xzvf libmodbus-3.0.x.tar.gz
```

Esto descomprime el archivo en una carpeta llamada "libmodbus-3.0.x"

Acto seguido se ingresa a esa carpeta con el siguiente comando:

```
cd libmodbus-3.0.x
```

Luego se debe ejecutar el archivo de configuración. En general se realiza con:

```
sudo ./configure
```

Los últimos pasos consisten en realizar el compilado y luego instalarlo. Se hace con:

```
sudo make
```

Aquí pedirá la contraseña de root y luego

```
sudo make install
```

Anexo E. Establecer una IP fija en la Raspberry Pi

Usar una IP estática puede ser muy útil en caso que se necesite acceder fácilmente a la Raspberry Pi sin necesidad de establecer la dirección IP cada vez que se inicia o reconecta a la red (Por ejemplo usando servicios como SSH, (S)FTP, etc.)

Para esto se necesita modificar el archivo /etc/network/interfaces

Antes de realizar alguna modificación se recomienda hacer una copia de respaldo del archivo interface en el caso de que ya se encuentre presente alguno.

```
pi@raspberry:sudo cp /etc/network/interfaces /etc/network/interfaces.sav
```

Se puede editar el archivo con cualquier editor de texto como vi o vim.
Se necesitan privilegios de administrador, para ello se utiliza sudo:

```
pi@raspberry:sudo vi /etc/network/interfaces
```

En el archivo *interfaces* se encuentra una línea como la siguiente:

```
iface eth0 inet dhcp
```

Esta línea habilita el cliente DHCP. Esta línea ya no va a seguir siendo usada por lo que se debe borrar o colocar un # al principio para comentarla, así:

```
#iface eth0 inet dhcp
```

En el archivo se deben insertar las siguientes líneas:

```
#dispositivo loopback
auto lo
iface lo inet loopback
#dispositivo eth0
auto eth0
iface eth0 inet static
#IP estatica
address 192.168.1.5
#Puerta de Enlace
gateway 192.168.1.1
netmask 255.255.255.0
#Datos de la familia de red
```

```
network 192.168.1.0
broadcast 192.168.1.255
```

Solamente la dirección IP y la puerta de enlace son estrictamente necesarias. Luego se necesita reiniciar la configuración de red.

```
pi@raspberry:sudo /etc/init.d/networking restart
```

Anexo F. Acceder a la Raspberry Pi a través de SSH y transferir archivos.

Una forma de utilizar la Raspberry con línea de comando es mediante SSH (intérprete de órdenes segura).

La instalación de Raspbian en la Raspberry Pi ya trae corriendo el demonio de SSH. Este estará escuchando por el puerto 22 alguna petición de conexión. Se puede usar línea de comando, transferencia de archivos o una interfaz gráfica de usuario (GUI) a través de la conexión del puerto 22.

Para la conexión física usando cable de red debe tenerse sumo cuidado en usar únicamente un cable de tipo cruzado.

Para que la conexión tenga éxito, por el lado del cliente, se debe tener un programa cliente de SSH. En algunas versiones de Linux este ya está instalado.

En caso de no estar instalado puede instalarse el servidor openssh-server con el siguiente comando.

```
$sudo apt-get install ssh
```

Primero debe de comprobarse que la tarjeta de red tiene una IP establecida o que el servicio de administración de red no esté cambiado la IP automáticamente.

Para detener el servicio administrador de red en Ubuntu o Debian puede usarse el siguiente comando.

```
~$ sudo service network-manager stop
```

O en el caso de otra distribución de Linux puede usarse el comando.

```
~$ sudo /etc/init.d/networking stop
```

Luego necesita levantarse el servicio manualmente para que no cambie la IP. Se utilizan los siguientes comandos.

```
~$ sudo ifconfig eth0 up
```

Donde eth0 es la interfaz de red que se está utilizando.

```
~$ sudo ifconfig eth0 192.168.1.6 netmask 255.255.255.0
```

Donde 192.168.1.6 es la IP elegida para la maquina con una máscara 255.255.255.0

Ahora se puede hacer ping para comprobar que ya existe conexión, 192.168.1.5 es la IP establecida anteriormente a la Raspberry Pi.

```
~$ ping 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
64 bytes from 192.168.1.5: icmp_req=1 ttl=64 time=1.16 ms
64 bytes from 192.168.1.5: icmp_req=2 ttl=64 time=0.507 ms
64 bytes from 192.168.1.5: icmp_req=3 ttl=64 time=0.599 ms
```

Para establecer la conexión se utiliza el siguiente comando.

```
ssh  -X <Dirección IP de la Raspberry> -l <Usuario de la Raspberry establecido anteriormente>
ssh -X 192.168.1.200 -l pi
```

Al iniciar por primera vez el servidor es posible que no reconozca la autenticidad del host, se escribe yes para continuar.

```
The authenticity of host '192.168.1.5 (192.168.1.5)' can't be established.
ECDSA key fingerprint is cc:aa:af:11:05:60:eb:c3:d0:9d:fe:3e:41:3c:48:84.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.5' (ECDSA) to the list of known hosts.
pi@192.168.1.5's password:

Linux raspberrypi 3.6.11+ #371 PREEMPT Thu Feb 7 16:31:35 GMT 2013 armv6l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 26 20:56:04 2013
pi@raspberrypi ~ $
```

El comando -X en la conexión mediante SSH me permite hacer uso de un servidor gráfico para mostrar la pantalla de Raspberry PI de manera remota.

Para abrir el navegador web de la Raspberry PI de manera remota, se puede utilizar el comando:

```
midori &
```

El “&” significa que midori estará corriendo en segundo plano para tener la consola de linux libre

F.1 Transferir archivos a través de SSH.

La transferencia de los archivos y ejecutables necesarios para la consulta de los medidores se realiza a través de SSH usando los siguientes comandos:

```
scp /home/name/<archivo> <usuario>@<dirección IP>:/home/name/<folder de destino>
```

Por ejemplo:

```
scp document.odt ubuntu@192.168.1.10:/home/ubuntu/Escritorio
```

Para transferir una carpeta completa mediante SSH se utiliza la siguiente sintaxis:

```
scp -r /home/name/<carpeta> <usuario>@<dirección IP>:/home/name/<folder de destino>
```

Anexo G. Cambiar zona horaria a Raspberry Pi

Por defecto la zona horaria de la Imagen de la Rasberry Pi es la del Reino Unido. Como la Raspberry No tiene un reloj de tiempo real, cuando la Raspberry se enciende y si se tiene conexión a Internet la hora se sincroniza. Para establecer la zona horaria de la zona locan donde se encuentre, se necesita cambiar la configuración del paquete *tzdata*.

```
sudo dpkg-reconfigure tzdata
```

En la Figura 34 se muestra la selección de territorio

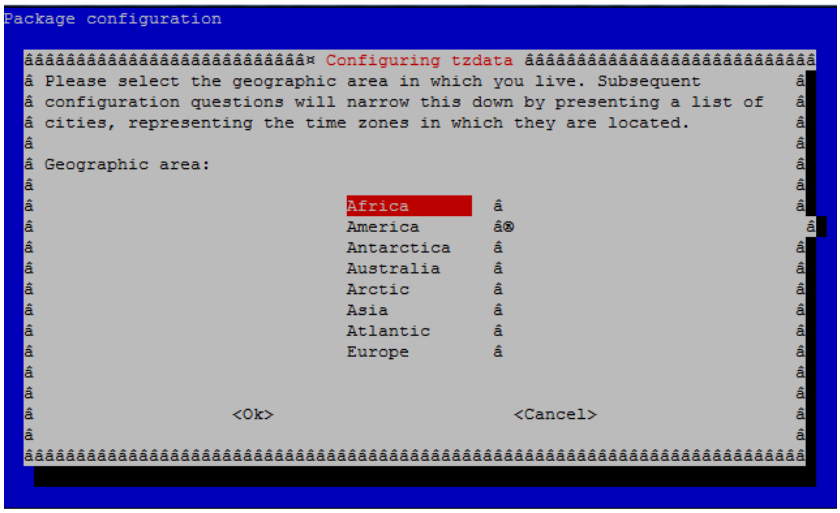


Figura 34 Configuración zona horaria - territorio Raspberry Pi.

Luego se selecciona la hora de la zona local, como se muestra en la Figura 35.

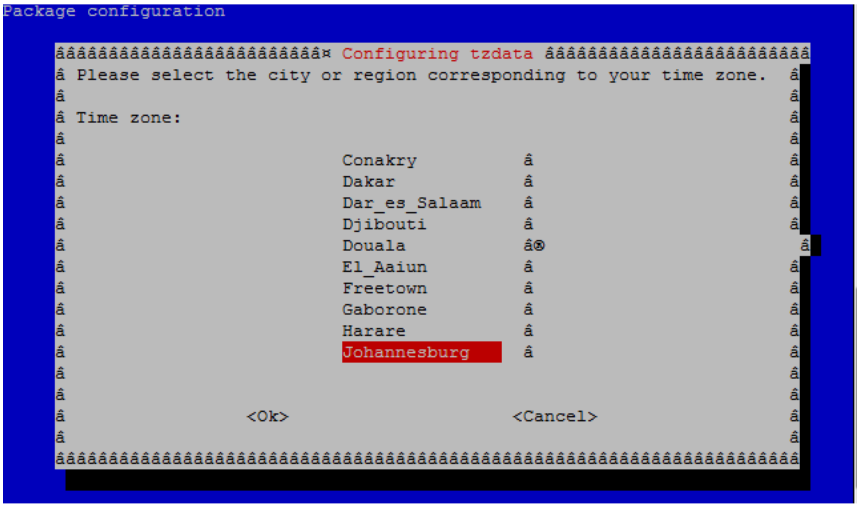


Figura 35 Selección de zona horaria Raspberrry Pi.

Anexo H. Instalación de la API de MySQL para lenguaje C

Para la utilización de MySQL desde código C se necesitan las librerías "*my_global.h"*, *"mysql.h m"* y *"my_sys.h*" estas se encuentran para Linux en el paquete: *libmysqlclient15-dev.* Para realizar la instalación de este paquete en Debian se puede hacer mediante la siguiente línea de comando:

```
sudo apt-get install libmysqlclient15-dev
```

Con la instalación de esta librería, se puede hacer uso mediante el comando `*mysql_config --cflags --libs*` por lo que la compilación para un caso en particular de un programa cliente seria la siguiente:

```
gcc agronomia.c -o agronomia `mysql_config --cflags --libs` `pkg-config --libs --cflags libmodbus`
```

Anexo I. Instalación de Apache, MySQL, PHP en sistema basado en Linux Debian.

I.1 Instalación Apache

Apache es un software open source que corre en más del 50% de los servidores web. Para instalar apache se utilizan los siguientes comandos desde una terminal de Linux

```
sudo apt-get update
sudo apt-get install apache2
```

Para revisar que apache está instalado, se necesita escribir la dirección IP del servidor, por ejemplo: http://127.0.1.0. La página debe de mostrar el aviso “It works!"

I.2 Encontrar la dirección IP del servidor

Se puede ejecutar el siguiente comando para encontrar la dirección IP del servidor apache:

```
ifconfig eth0 | grep inet | awk '{ print $2 }'
```

I.3 Instalación MySQL

MySQL es un poderoso gestor de base de datos usado para organizar y recuperar datos.

Para instalar MySQL en sistema Linux basado en Debian, se escribe en terminal el siguiente comando:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

Durante la instalación, MySQL pedirá que se establezca la contraseña de super usuario. Si se pierde la oportunidad de establecer la contraseña durante la instalación del programa, es fácil poderla establecer luego con la terminal de MySQL.

Una vez se ha instalado MySQL, se debe activar con los siguientes comandos:

```
sudo mysql_install_db
```

La instalación finaliza ejecutando el siguiente script MySQL:

```
sudo /usr/bin/mysql_secure_installation
```

La terminal pedirá por la contraseña de super usuario actual.

Luego la terminal preguntará si se desea cambiar la contraseña de super usuario. Se debe elegir N y seguir con el siguiente paso.

Por fines prácticos se selecciona “Yes” a todas las opciones. Al final, MySQL se configura e implementará los nuevos cambios.

```
By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into
production environment.
```

```
Remove anonymous users? [Y/n] y
 ... Success!
```

```
Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n] y
... Success!
```

```
By default, MySQL comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.
```

```
Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.
```

```
Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...
```

I.4 Instalación PHP

PHP es un lenguaje de programación web libre que es ampliamente utilizado para construir páginas web dinámicas.

Para instalar PHP, se ejecuta el siguiente comando en una terminal:

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Luego se agrega php en el índice de directorio, para esto se edita el siguiente archivo

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Se agrega "*index.php*" al principio del archivo index. La pagina debe de verse como esta:

```
<IfModule mod_dir.c>

        DirectoryIndex index.php index.html index.cgi index.pl index.php index.xhtml index.htm

</IfModule>
```

I.5 Instalación de Módulos PHP

PHP también tiene una variedad de útiles librerías y módulos que se pueden agregar al servidor. Se pueden ver las librerías disponibles con el siguiente comando:

```
apt-cache search php5-
```

La terminal desplegará una serie de posibles módulos, una vez se decida instalar el módulo se debe escribir:

```
sudo apt-get install <name of the module>
```

Se pueden instalar múltiples librerías de una vez separando con un espacio los nombres de cada módulo.

I.6 Revisar el servidor funcionando

Aunque PHP ya esta instalado, se puede revisar y ver los componentes en línea creando rápidamente una página de información de php

Para crear la página, primero se crea un nuevo archivo:

```
sudo nano /var/www/info.php
```

Y se agregan las siguientes líneas:

```
<?php
phpinfo();
?>
```

Luego guardar y salir

Reiniciar apache para que los cambios surjan efecto:

```
sudo service apache2 restart
```

Para finalizar se puede ver la página de información recién creada accediendo a :

```
http://<direccion ip>info.php
```

Mostrará una página similar a esta:

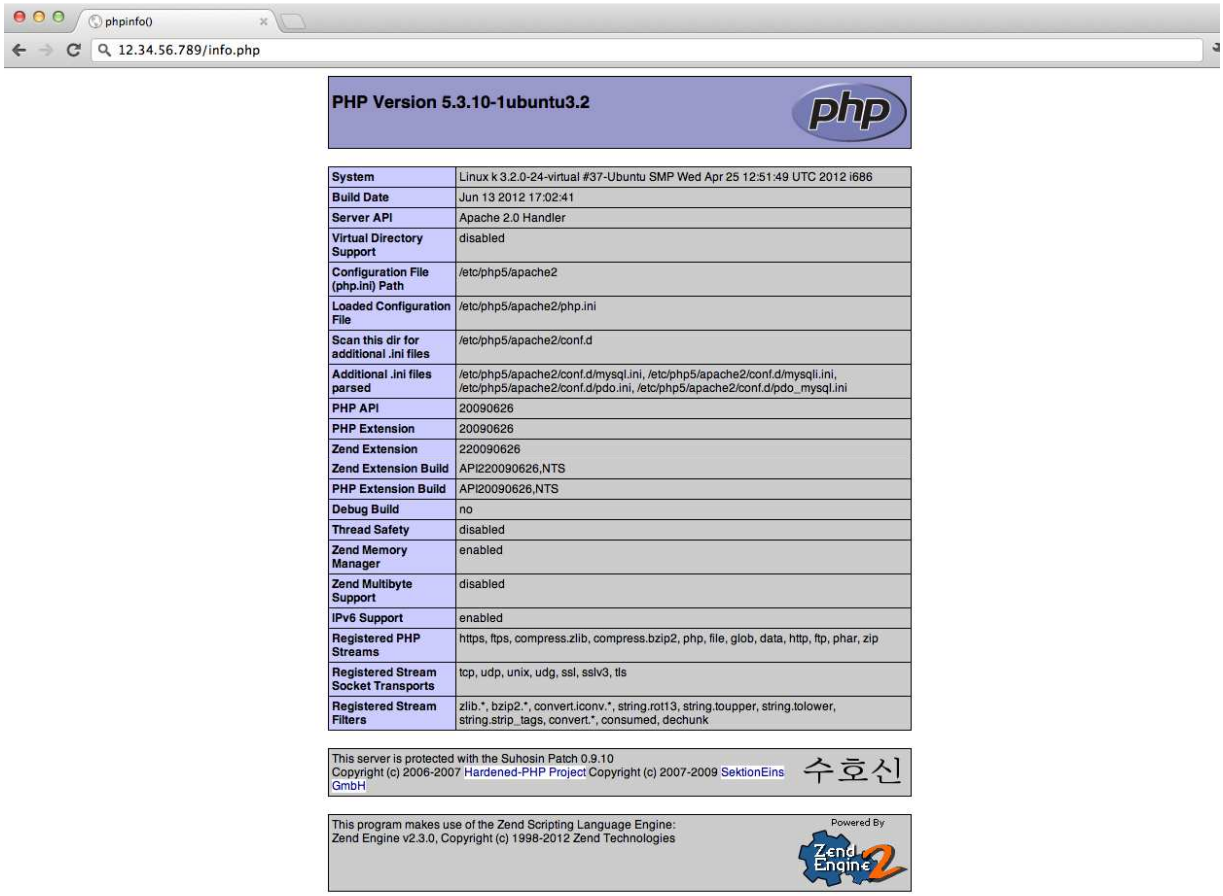


Figura 36 PHP ejecutándose correctamente luego de su instalación.

Anexo J. Instalación segura de phpMyAdmin para manipulación de bases de datos mediante navegador web

J.1 Acerca de phpMyAdmin

phpMyAdmin es un software web libre para trabajar con MySQL en la web. Proveer un diseño frontal muy conveniente para las características que MySQL posee.

J.2 Configuración

Antes de trabajar con phpMyAdmin se necesita tener instalado LAMP en el servidor de datos o en la misma máquina desde donde se realizarán las peticiones.

J.3 Instalación phpMyAdmin

La manera más fácil de instalar phpmyadmin es a través de apt-get, para los sistemas basados en Debian.

```
sudo apt-get install phpmyadmin
```

Durante la instalación, phpMyAdmin lo guiará por una configuración básica , una vez el proceso comienza, sigua los siguientes pasos:

- Selecciones Apache2 para el servidor
- Elija YES cuando se pregunte sobre si configurar la base de datos para phpMyAdmin con dbconfig-common
- Ingrese su MySQL password cuando se le indique
- Ingrese el password que usted quiere usar para ingresar a phpMyAdmin

Después que la instalación ha terminado, agregue phpMyAdmin a la configuración de Apache

```
sudo nano /etc/apache2/apache2.conf
```

Agregue la configuración phpMyAdmin al archivo

```
Include /etc/phpmyadmin/apache.conf
```

Reinicie apache:

```
sudo service apache2 restart
```

Puede acceder a phpMyAdmin con la ip del servidor/phpmyadmin desde cualquier navegador web. Por Ejemplo http://127.0.1.1/phpmyadmin

J.4 Seguridad de phpMyAdmin

Desafortunadamente las versiones anteriores de phpMyAdmin tenían serias vulnerabilidades de seguridad. Se pueden prevenir la mayoría de ataques a través de un simple proceso: bloqueando el directorio completo con las restricciones de usuario y password nativo de Apache, que evitara que usuarios remotos intenten explotar vulnerabilidades de versiones anteriores de phpMyAdmin.

J.5 Cambiar la configuración de PhpMyAdmin

El primer paso es cambiar la configuración de PhpMyAdmin, para ello se modifica el siguiente archivo:

```
sudo nano etc/phpmyadmin/apache.conf
```

Agregue el comando *'AllowOverride All'* bajo la línea **'DirectoryIndex'** debe quedar algo como esto:

```
<Directory /usr/share/phpmyadmin>
    Options FollowSymLinks
    DirectoryIndex index.php
    AllowOverride All
```

J.6 Configurar el archivo .htaccess

El siguiente paso es configurar el archivo.htaccess. Con acceso al archivo .htaccess podemos proceder a configurar la creación de un usuario nativo cuyo inicio de sesión que se requeriría para incluso acceder a la página de entrada phpmyadmin.

Se comienza por crear la página .htaccess en el directorio phpmyadmin:

```
sudo nano /usr/share/phpmyadmin/.htaccess
```

A continuación se realiza la configuración del usuario en el archivo .htaccess. Se utiliza el siguiente texto:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile //usr/share/phpmyadmin/.htpasswd
Require valid-user
```

A continuación se presenta una breve explicación de cada línea:

- **AuthType:** Este se refiere al tipo de autenticación que será utilizada para verificar las contraseñas. Las contraseñas son verificadas a través de HTTP y la palabra básico no debe de ser cambiada
- **AuthName:** Este será el texto que será desplegado a la hora de solicitar la contraseña. Se puede poner cualquier frase
- **AuthUserFile:** Esta línea asigna la dirección del server al archivo con la contraseña (el cual será creado en el siguiente paso).
- **Require valid-user:** Esta línea le dice al archivo .htaccess que solamente los usuarios definidos en el archivo de contraseñas pueden ingresar a la página de acceso de phpMyAdmin.

J.7 Crear el archivo htpasswd.

Ahora se procede a crear la información del usuario válida. Comenzando con crear el archivo htpasswd, se utiliza el comando htpasswd, y se coloca el archivo en alguna carpeta que no sea accesible desde el navegador. Aunque se puede nombrar el archivo con cualquier nombre, lo conveniente es llamarlo .htpasswd. Por ejemplo:

```
sudo htpasswd -c  /usr/share/phpmyadmin/.htpasswd root
```

-c = crea un nuevo archivo
/usr/share/phpmyadmin/ = lugar donde se colocará el archivo .htpassws
root = nombre del usuario

Un cursor le pedirá que provea y confirme la contraseña. Una vez el usuario y el password son guardados. se puede ver que la contraseña está encriptada en el archivo.

Se termina reiniciando apache:

```
sudo service apache2 restart
```

J.8 Acceder a phpMyAdmin

PhpMyAdmin estara mucho mas seguro ahora ya que solamente usuarios autorizados tendrán acceso a la página de acceso. Accediendo con <direccion_ip>/phpmyadmin deberá mostrar una imagen como la siguiente:

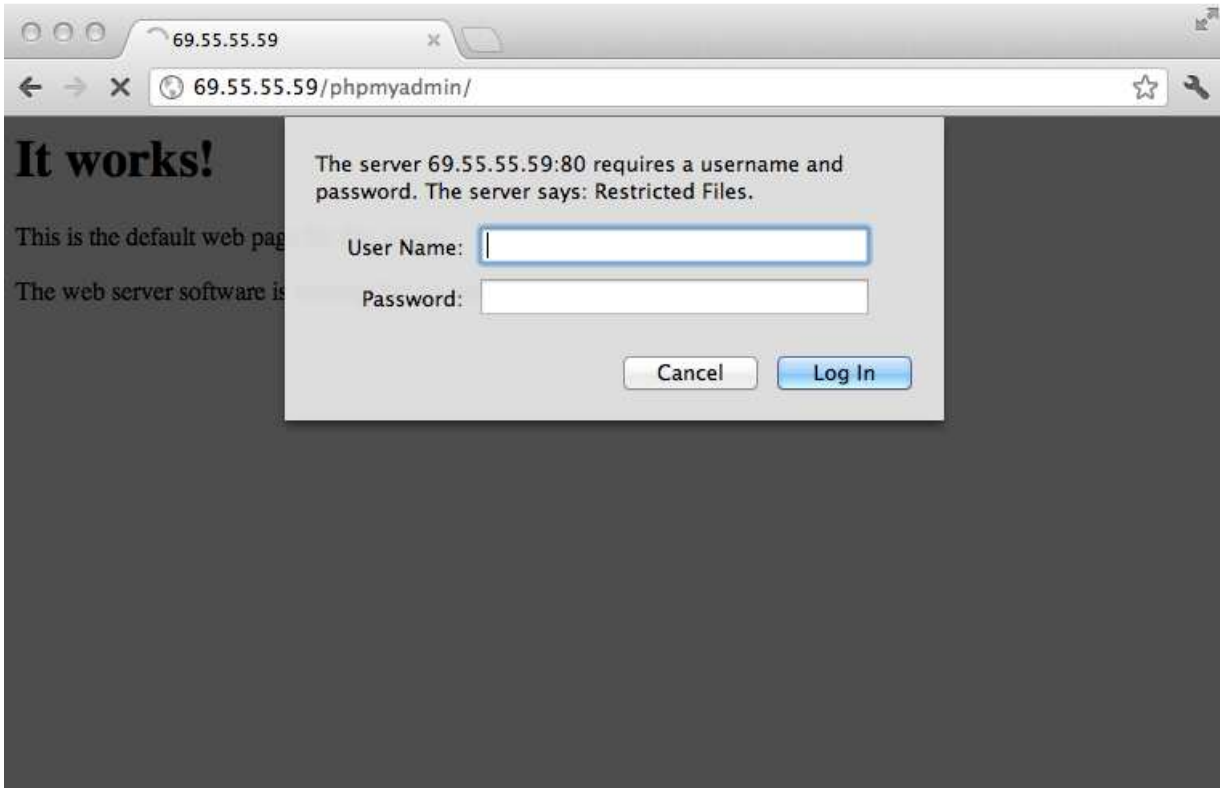


Figura 37 phpMyAdmin instalado de manera segura.

Se ingresa con el usuario y el password generado. Luego de entrar se puede acceder a phpMyAdmin con el usuario y contraseña de MySQL

Anexo K. Descripción código de sincronización del reloj de los medidores Shark 200-S

Los medidores Shark 200S cuenta con un reloj de tiempo real que puede ser programado con el *software* original para la plataforma Windows brindado por el fabricante o a través del protocolo Modbus en cualquier sistema operativo. Al interrogar la hora de los medidores Shark 200-S se descubrió que estos tenían un par de años sin ser actualizados. Para ello es necesario hacer uso de la función *modbus_write_register()* la cual escribe en la memoria del medidor, cambiando así la hora del mismo.

K.1 Diseño del programa

La sincronización de la hora de los medidores se realiza escribiendo en la memoria asignada para el reloj del medidor la hora actual de la máquina donde se está corriendo el programa, esta máquina previamente ha obtenido la hora de un servidor de tiempo en internet. Para verificar la hora de los medidores se ha agregado una columna con la hora de los medidores en la base de datos de MySQL, que esta contiguo a la columna de la hora de la computadora. Por simplicidad en la implementación del código se verifica manualmente si hay una desincronización entre las dos columnas de los tiempos y en caso de ser necesaria la sincronización, se ejecutará manualmente el código desarrollado.

La tarea de actualización de la hora no está obligada a cumplirse en un tiempo determinado, por lo tanto, la actualización se hace cambiando la hora del medidor uno después de otro.

K.2 Descripción del código

Para mantener un diseño más ordenado, el código se compone de dos archivos, el archivo main que contiene las direcciones de todos los medidores a actualizar y el archivo de configuración que contiene la función de actualización.

```
//*** hora_update.c ***

//Compilar asi: gcc hora_update.c tupdate200.c -o hora_update `pkg-config --libs --cflags libmodbus`

#include <stdio.h>
#include <unistd.h>
```

```
#include <stdlib.h>
#include <errno.h>
#include <modbus.h>
#include <string.h>
#include <time.h>
#include <sys/time.h>
#include "tupdate200.h"

int main(int argc, char *argv[])
{

hora_update_200("10.30.201.2", "Derecho");
//hora_update_200("10.30.206.2", "Economia5");
hora_update_200("10.30.214.2", "PrimarioFIA");
hora_update_200("10.30.221.2", "Quimica");
hora_update_200("10.30.224.2", "Odontologia1");
hora_update_200("10.30.225.2", "Odontologia2");
hora_update_200("10.30.226.2", "Odontologia3");
hora_update_200("10.30.227.2", "Medicina");
//hora_update_200("10.30.229.2", "Rectoria");
//hora_update_200("10.30.230.2", "Artes");

return 0;

}
```

El código anterior únicamente hace llamada a la función hora_update_200() con el argumento de IP e Identificación de cada medidor, cada llamada a función se realizará hasta que la llamada previa a esa haya terminado, esto significa que en caso que el enlace de la red este débil, la función tendrá que esperar hasta finalizar correctamente para luego volver a ser llamada con los parámetros del siguiente medidor.

```
//*** tupdate200.c ***

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <modbus.h>
#include <string.h>
```

```
#include <time.h>
#include <sys/time.h>
#include "tupdate200.h"

#define SERVIDOR 0x01    //Direccion asignada al servidor

int hora_update_200(char *ip, char *id) /*Definicion de la funcion*/
{
  modbus_t *ctx;            //Crea puntero

uint16_t tab_reg[6];
uint16_t tab_rq_registers[2];
uint16_t toFloat[2];
int rc;
int lee_fecha[6];
int puerto=502;
int nb;

  struct timeval old_response_timeout;
  struct timeval response_timeout;
printf("\n          conectando con %s (%s):\n",id,ip);
/* Establecer un contexto MODBUS */
  ctx = modbus_new_tcp(ip, puerto); //Usando contexto TCP, ip y puerto
 if (ctx == NULL) {
          fprintf(stderr, "No pudo ubicar el contexto libmodbus\n");
          return -1;
    }

/* Establece el numero del Slave en el contexto TCP */
  modbus_set_slave(ctx, SERVIDOR);

/* Establece el tiempo de espera para cada respuesta*/
  response_timeout.tv_sec = 59;
  response_timeout.tv_usec = 0;
  modbus_set_response_timeout(ctx,&response_timeout);

/* Establece la conexion MODBUS */
if (modbus_connect(ctx) == -1) {
          fprintf(stderr, "La conexion fallo: %s\n", modbus_strerror(errno));
          modbus_free(ctx);
          return -1;
    }
```

```
/* Hace la peticion de la hora del medidor  */
  rc = modbus_read_registers(ctx, 0x119b, 3,tab_reg);

if (rc == -1) {
          fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
          return -1;
          }

          MODBUS_SET_INT16_TO_INT8(lee_fecha, 0,tab_reg[0]);
          MODBUS_SET_INT16_TO_INT8(lee_fecha, 2,tab_reg[1]);
          MODBUS_SET_INT16_TO_INT8(lee_fecha, 4,tab_reg[2]);
          printf(" Hora Actual del Medidor: 20%d-%d-%d
%d:%d:%d\n",(int)lee_fecha[0],(int)lee_fecha[1],(int)lee_fecha[2],(int)lee_fecha[3],(int)lee_fecha[4
],(int)lee_fecha[5]);

/* Bloque de Actualizacion de Hora */

          int8_t fecha[6];

          time_t now=time(NULL);
          struct tm *tm=localtime(&now);
          fecha[0] = tm->tm_year - 100;
          fecha[1] = tm->tm_mon + 1;
          fecha[2] = tm->tm_mday;
          fecha[3] = tm->tm_hour;
          fecha[4] = tm->tm_min;
          fecha[5] = tm->tm_sec;
          printf("            Hora Internet: 20%d-%d-%d
%d:%d:%d\n",fecha[0],fecha[1],fecha[2],fecha[3],fecha[4],fecha[5]);

          int16_t set_fecha[3];
          set_fecha[0] = MODBUS_GET_INT16_FROM_INT8(fecha,0);
          set_fecha[1] = MODBUS_GET_INT16_FROM_INT8(fecha,2);
          set_fecha[2] = MODBUS_GET_INT16_FROM_INT8(fecha,4);


/* Abre modo de configuración del medidor */
          uint16_t pass[2];
          pass[0]=5555;
          int rcpass = modbus_write_register(ctx, 0x5209, pass[0]);
          if (pass[0] == -1) {
```

```
        fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
        return -1;
        }
/* Actualiza la hora del medidor */
        int wrc = modbus_write_registers(ctx, 0x520f, 3,set_fecha);
        if (wrc == -1) {
        fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
        return -1;
        }

/* Hace una nueva peticion de la hora del medidor para verificar si se ha actualizado  */
 rc = modbus_read_registers(ctx, 0x119b, 3,tab_reg);

if (rc == -1) {
        fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
        return -1;
        }

        MODBUS_SET_INT16_TO_INT8(lee_fecha, 0,tab_reg[0]);
        MODBUS_SET_INT16_TO_INT8(lee_fecha, 2,tab_reg[1]);
        MODBUS_SET_INT16_TO_INT8(lee_fecha, 4,tab_reg[2]);
        printf("Hora Actualizada del Medidor: 20%d-%d-%d
%d:%d:%d\n",(int)lee_fecha[0],(int)lee_fecha[1],(int)lee_fecha[2],(int)lee_fecha[3],(int)lee_fecha[4
],(int)lee_fecha[5]);

 /* Cierra la conexion MODBUS */
 modbus_close(ctx);
 modbus_free(ctx);
return(0);
}
```

La cabecera del programa es muy similar a la del código de petición de datos con la diferencia de los siguientes bloques:

```
/* Peticion de la hora del medidor  */

 rc = modbus_read_registers(ctx, 0x119b, 3,tab_reg);

if (rc == -1) {
        fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
        return -1;
```

```
        }

        MODBUS_SET_INT16_TO_INT8(lee_fecha, 0,tab_reg[0]);
        MODBUS_SET_INT16_TO_INT8(lee_fecha, 2,tab_reg[1]);
        MODBUS_SET_INT16_TO_INT8(lee_fecha, 4,tab_reg[2]);
        printf(" Hora Actual del Medidor: 20%d-%d-%d
%d:%d:%d\n",(int)lee_fecha[0],(int)lee_fecha[1],(int)lee_fecha[2],(int)lee_fecha[3],(int)lee_fecha[4
],(int)lee_fecha[5]);
```

En este bloque se hace la petición de datos a los 3 registro Hexadecimales consecutivos, comenzando del registro 0x119b dichos registros tiene el formato siguiente.

Byte	0	1	2	3	4	5
Value	Year	Month	Day	Hour	Minute	Second
Range	0-99 (+2000)	1-12	1-31	0-23	0-59	0-59
Mask	0x7F	0x0F	0x1F	0x1F	0x3F	0x3F

Tabla 15 Registros para el reloj de tiempo real de los Shark 200-S

Siendo el Byte 0 y 1 el registro 1, Byte 2 y 3, registro 2, Byte 4 y 5 registro 3. Para separar los dos bytes de manera individual y enmascararse para obtener la hora correcta, se hace uso de la función MODBUS_SET_INT16_TO_INT8(lee_fecha, 0,tab_reg[0]); siendo lee_fecha un arreglo de 6 x 1 Bytes , guardando en dos registros consecutivos comenzando desde la posición 0, y tab_reg[0] arreglo de 3 x 2 Bytes comenzando desde la posición 0 del registro de 2 Bytes.

Para imprimir se enmascara con +2000 el primer Byte leído, ya que como muestra la tabla la cuenta disponible en ese Byte es de 0-99, quedando como *20%d* tal como se muestra en el código.
Para convertir directamente de un tipo hexadecimal a entero se hace uso de una conversión explícita del lenguaje C denominada *"cast"* por ejemplo *(int)lee_fecha[1]* convierte del tipo origen, en este caso hexadecimal al tipo entero, reduciendo en gran medida la tarea de convertir los datos.

```
/* Bloque de Actualizacion de Hora */
```

```
        int8_t fecha[6];

        time_t now=time(NULL);
        struct tm *tm=localtime(&now);
        fecha[0] = tm->tm_year - 100;
        fecha[1] = tm->tm_mon + 1;
        fecha[2] = tm->tm_mday;
        fecha[3] = tm->tm_hour;
        fecha[4] = tm->tm_min;
        fecha[5] = tm->tm_sec;
        printf("          Hora Internet: 20%d-%d-%d
%d:%d:%d\n",fecha[0],fecha[1],fecha[2],fecha[3],fecha[4],fecha[5]);

        int16_t set_fecha[3];
        set_fecha[0] = MODBUS_GET_INT16_FROM_INT8(fecha,0);
        set_fecha[1] = MODBUS_GET_INT16_FROM_INT8(fecha,2);
        set_fecha[2] = MODBUS_GET_INT16_FROM_INT8(fecha,4);
```

Para extraer la hora de la computadora se utiliza la libreria <time.h> donde se declara la variable now del tipo time_t . Utilizando la estructura tm, que pertenece a la libreria <time.h> se accede a los diferentes datos de la hora y la fecha, copiandolos al arreglo fecha[]. Para comprobar que el arreglo fecha ha sido copiado correctamente, se manda a imprimir en pantalla. Luego se trasladan esos datos pasandolos de 1 Byte a registros de 2 Bytes.

```
/* Abre modo de configuración del medidor */
        uint16_t pass[2];
        pass[0]=5555;
        int rcpass = modbus_write_register(ctx, 0x5209, pass[0]);
        if (pass[0] == -1) {
        fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
        return -1;
        }
```

El bloque abre modo de configuración, se ingresa el password del medidor usando la función *modbus_write_register(ctx, 0x5209, pass[0]);* dirigida a la dirección de memoria *0x5209* la cual es la especificada para recibir dicho password, esto para poder entrar al modo de configuración. con lo que podremos modificar los registros del medidor, de lo contrario el medidor recibe los registros pero no los guarda.

```
/* Actualiza la hora del medidor */

        int wrc = modbus_write_registers(ctx, 0x520f, 3,set_fecha);
        if (wrc == -1) {
        fprintf(stderr, "Error en la adquisicion de los datos  %s\n", modbus_strerror(errno));
        return -1;
        }
```

Se escriben los registros ya guardados en el arreglo *set_fecha* a la dirección en *0x520f*

Finalmente se vuelve a leer el medidor para comprobar si efectivamente se actualizo la hora.

Anexo L. Rendimiento de Raspberry Pi como servidor LAMP

L.1 Temperatura de funcionamiento

El funcionamiento continuo y sin interrupciones de la microcomputadora Raspberry Pi desempeña un papel fundamental en la recolección de datos de la red, ya que algún fallo en la alimentación y o en el sistema operativo afectaría la continuidad de los valores instantáneos leídos de la red, mas no así en los valores de energía y demanda, debido a que los datos de energía y demanda son acumulados dentro de los medidores. Por lo que fue necesario poner a prueba el rendimiento del equipo durante un tiempo prolongado.

Durante la primera puesta a prueba se utilizó la configuración por defecto la cual consiste en colocar el hardware en un case acrílico como protección ambiental.

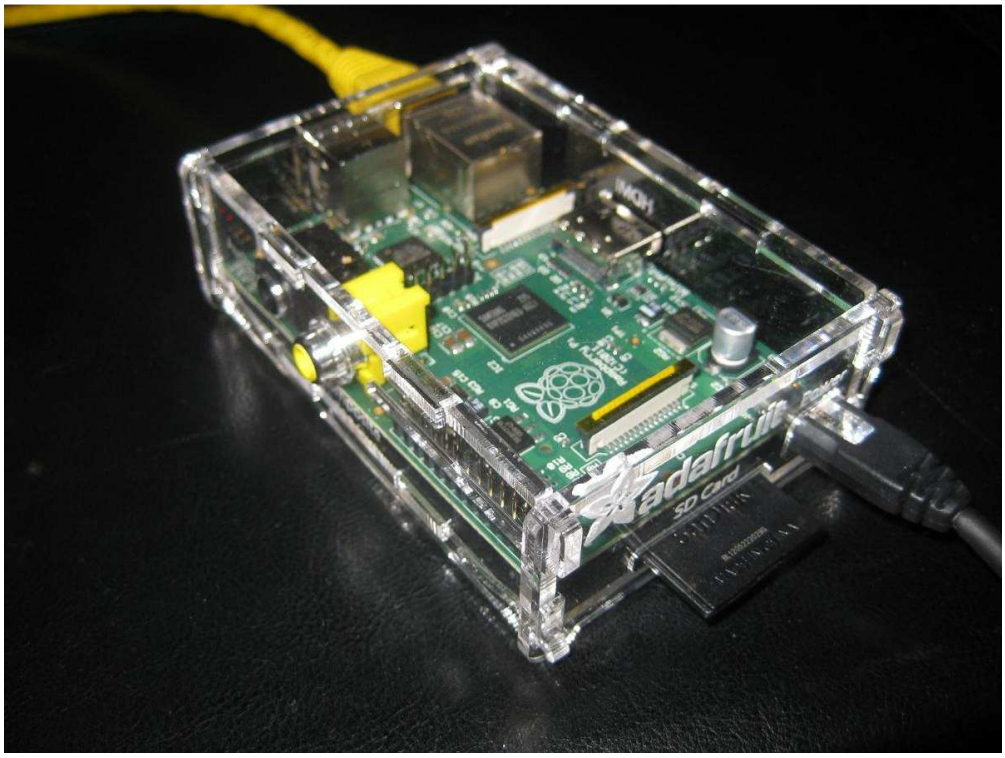


Figura 38 Raspberry Pi en case acrílico.

Una de las ventajas de utilizar la micro computadora Raspberry Pi es que cuenta con dos sensores de temperatura internos. Uno para la CPU y otro para la GPU permitiendo monitorear la temperatura de operación.

El resultado de trabajar con la temperatura ambiente de nuestro país dentro del case acrílico, llevo a operar la computadora a no menos de 65°c lo cual se encuentra dentro de los límites de operación segura del hardware.

Una investigación realizada por el sitio web www.geektopia.es sobre la temperatura de operación arrojo los siguientes resultados:

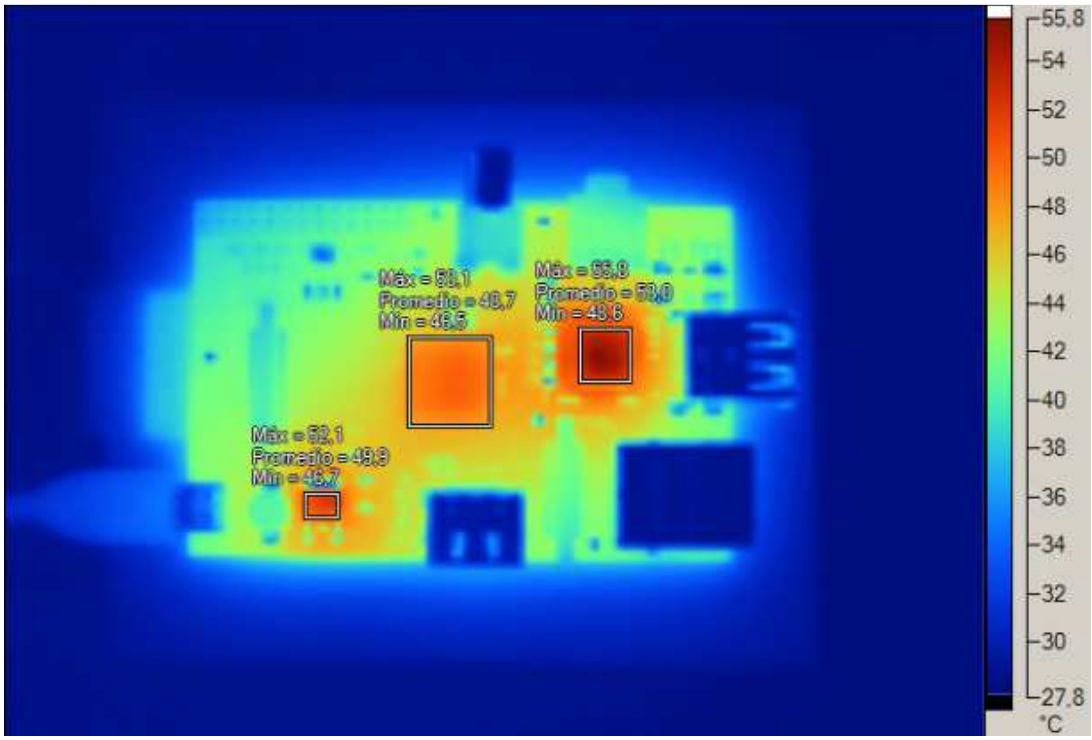


Figura 39 Temperatura de funcionamiento Raspberry Pi

Los valores de temperatura promedio en operación normal fueron 49,9ºC para el regulador de tensión, 48,7ºC para el SoC BCM2835 y 53ºC para el controlador de red LAN9512 bajo una temperatura ambiente que osciló entre los 26,7ºC al inicio de la prueba y 26,9ºC al final de la misma. Este estudio demostró que, para nuestro ambiente cálido, los rangos de operación llevarían al hardware a trabajar en sus límites seguros de operación en forma permanente, aun sin aumentar la carga de procesamiento de la computadora.

Ya dentro de las pruebas realizadas localmente, luego de tres días de operación continua a temperaturas de 65°C produjo un fallo en la memoria MicroSD, donde se ejecuta el sistema operativo corrompiendo el sistema de archivos y perdiendo los datos recolectados hasta el momento. Esto debido a que por su tamaño la tarjeta MicroSD se ubica muy cerca de procesador y la memoria RAM.

Luego de dicha experiencia se decidió instalar un ventilador para reducir al máximo la temperatura de operación.

Para ello se optó por una caja 5x5 de instalaciones eléctricas y un ventilador de fuente de poder ATX para PC, el cual se ajusta muy bien con la ubicación de los tornillos de la caja 5x5.

Luego de la instalación del ventilador la operación de la computadora no pasa más de 33 grados como se muestra a continuación.

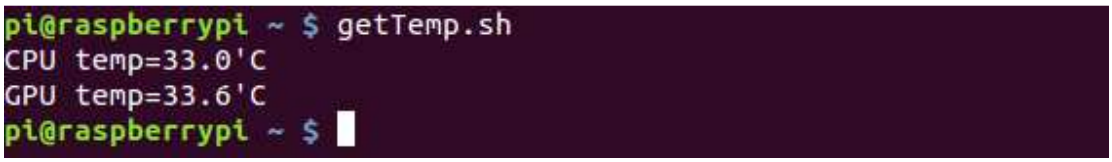


Figura 40 Temperatura de operación, luego de instalación de ventilador.

El script utilizado para obtener la temperatura del CPU y la GPU es el siguiente:

```
#!/bin/bash
cpuTemp0=$(cat /sys/class/thermal/thermal_zone0/temp)
cpuTemp1=$(($cpuTemp0/1000))
cpuTemp2=$(($cpuTemp0/100))
cpuTempM=$(($cpuTemp2 % $cpuTemp1))

echo CPU temp"="$cpuTemp1"."$cpuTempM"'C"
echo GPU $(/opt/vc/bin/vcgencmd measure_temp)
```

L.2 Estabilidad del sistema

Una vez que la temperatura del hardware es controlada, el sistema operativo es muy estable ya que se encuentra corriendo un sistema Linux basado en Debian 7.

La imagen a continuación muestra la cantidad de días que el sistema tiene sin ningún reboot.

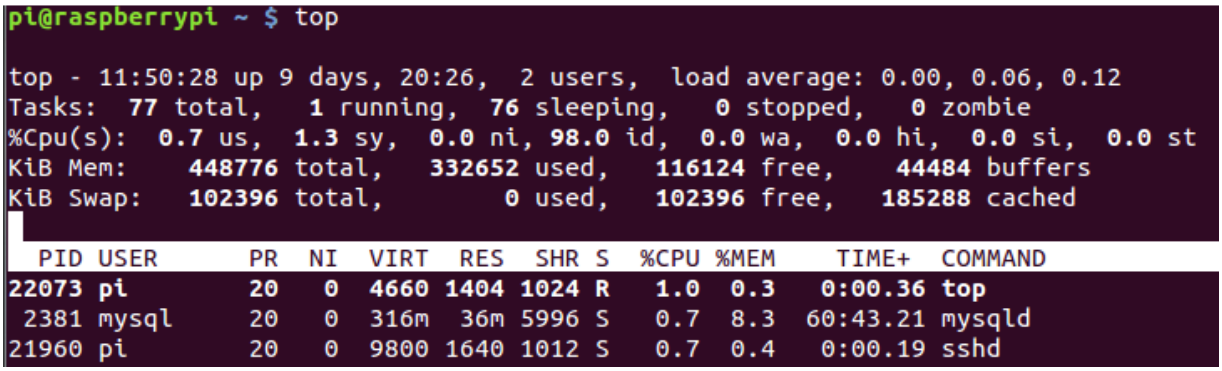


Figura 41 Tiempo de ejecución del sistema operativo sin ningún reinicio.

Para poder monitorear las veces en las que el sistema se ha reiniciado se implemento un script que deja en un archivo .log cada uno de los eventos. El script es el siguiente:

```
#!/bin/sh
# deja un log sobre cuando se reinició el sistema

date
echo El sistema ha iniciado
echo ----------------------------
```

Éste script ha sido agregado al programa crontab con la siguiente sintaxis:

```
@reboot cd /ubicacion donde se encuentra el archivo && ./logboot.sh >> /ubicacion donde se encuenta el archivo log/log_reboot.log 2>&1
```

@reboot indica que el script será ejecutado cada vez que el sistema se reinicie
>> indica que el resultado del script se añadirá en el archivo log.

Anexo M. Instalación de un Real Time Clock (RTC) a Raspberry Pi

Para poder mantener los costos bajos, la Raspberry Pi no incluye un módulo de Real Time Clock. En vez de ello, los usuarios deben de estar siempre conectados a una red ethernet o WiFi con conexión a internet y sincronizarse con un servidor NTP. Para la red de medidores, cada vez que se realiza una petición se necesita tener la hora exacta, pero podría suceder que la red de medidores esté funcionando bien pero no haya conexión a internet para que el sistema actualice la hora, en el caso que suceda un reinicio del sistema.

Es por ello que se procedió a incluir un módulo externo de RTC. Se utilizó el módulo RTC que utiliza el integrado DS1307.

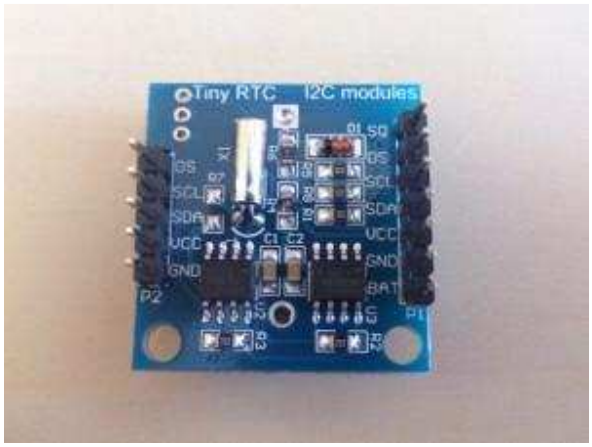


Figura 42 Módulo RTC.

Este módulo incluye todo los elementos necesarios para trabajar como una RTC, Tiene una salida de tipo I2C por lo que solamente debe de conectarse a los pines correctos en la Raspberry Pi.

M.1 Conexión RTC a Raspberry Pi

La conexión es la siguiente:

1. Conectar **VCC** del modulo al pin de **5.0V** de la Pi
2. Conectar **GND** de modulo al pin **GND** de la Pi
3. Conectar **SDA** de modulo al pin **SDA0** de la Pi
4. Conectar **SCL** de modulo al pin **SCL0** de la Pi

Los pines de salida del módulo RTC son los que se muestran en la siguiente imagen:

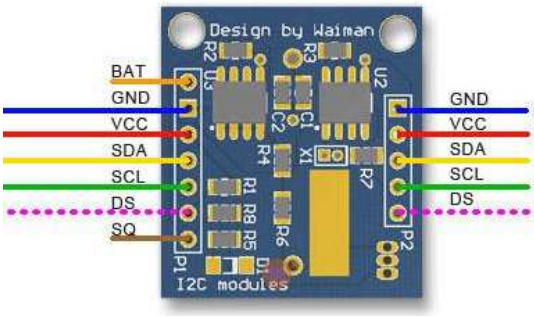


Figura 43 Pines de salida de la RTC.

Raspberry Pi cuenta con dos versiones y dos revisiones del hardware, cambiando la ubicación de los pines digitales entre diferentes versiones. Por lo que primero se debe determinar cuál es la versión y la revisión de hardware que se está utilizando. Desafortunadamente todas se ven muy similares, y puede ser difícil determinar cual es cual a simple vista, para ello existe un método usando la siguiente línea de comando.

```
cat /proc/cpuinfo
```

Con el cual se debe de obtener información como la siguiente:

```
Processor : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS : 697.95
Features : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part : 0xb76
CPU revision : 7
Hardware : BCM2708
Revision : 000f
Serial : 00000000d298cfc1
```

Esto nos muestra la revision code (0x0f) de placa, y a partir de ese codigo, y el excelente trabajo que Frank Caver en su blog http://raspberryalphaomega.org.uk quien ha realizado una tabla con todas las diferentes variables de placas de Raspberry Pi se puedo determinar que la placa utilizada es Revisión 2 Modelo B con; 512MB RAM, Ethernet, dos USB sockets, cinco LEDs, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5) . La Tabla 16 muestra la información completa según su número de revisión.

Rev. No.	Model	Capabilities
0x2	B1	Original Model B, 256MB RAM, Ethernet, two USB sockets, five LEDs, (P2) JTAG pins, no mounting holes, Pin3=GPIO0, Pin5=GPIO1, Pin13=GPIO21, I2C-0
0x3	B1+	Original Model B with no polyfuses, 256MB RAM, Ethernet, two USB sockets, five LEDs, no mounting holes, Pin3=GPIO0, Pin5=GPIO1, Pin13=GPIO21, I2C-0
0x4	B2	Model B, 256MB RAM, Ethernet, two USB sockets, five LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5)
0x5	B2	Model B, 256MB RAM, Ethernet, two USB sockets, five LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5)
0x6	B2	Model B, 256MB RAM, Ethernet, two USB sockets, five LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5)
0x7	A	Model A, 256MB RAM, no Ethernet, one USB socket, two LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, I2C-1, 8 extra IO pads (P5)
0x8	A	Model A, 256MB RAM, no Ethernet, one USB socket, two LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, I2C-1, 8 extra IO pads (P5)
0x9	A	Model A, 256MB RAM, no Ethernet, one USB socket, two LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, I2C-1, 8 extra IO pads (P5)
0xd	B2	Rev2 Model B, 512MB RAM, Ethernet, two USB sockets, five LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5)
0xe	B2	Rev2 Model B, 512MB RAM, Ethernet, two USB sockets, five LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5)

0xf	B2	Rev2 Model B, 512MB RAM, Ethernet, two USB sockets, five LEDs, mounting holes, Pin3=GPIO1, Pin5=GPIO2, Pin13=GPIO27, 12C-1, 8 extra IO pads (P5)

Tabla 16 Diferentes versiones de hardware según su número hexadecimal.

Tabla 16 sobre las diferentes revisiones de placa de Raspberry Pi y su modelo especifico []. tabla cortesia de Fran Carver en http://raspberryalphaomega.org.uk

Una vez determinado el modelo y la versión de la placa podemos determinar la ubicación de los pines I2C. El siguiente diagrama muestra gráficamente la ubicación de los pines I2C.

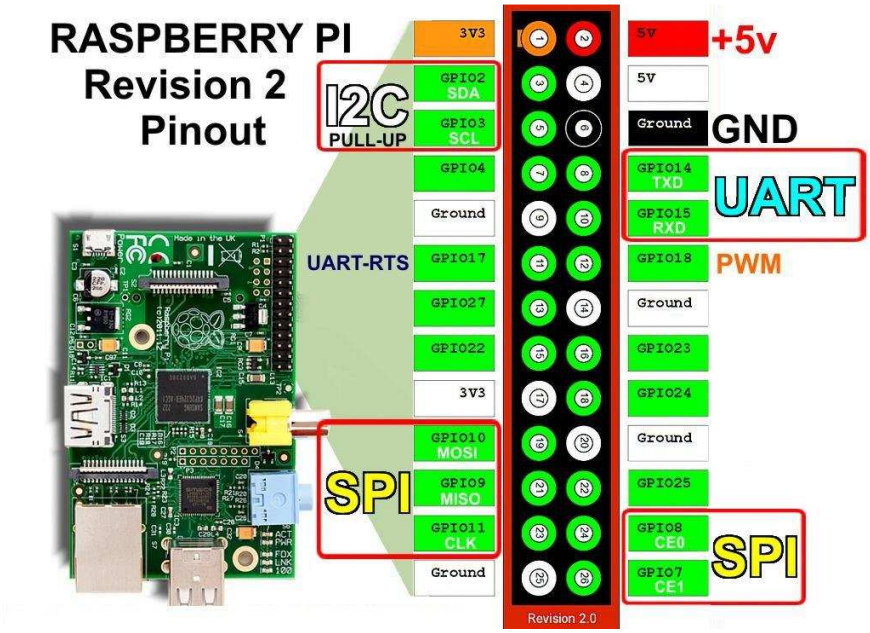


Figura 44 Ubicación de los pines I2C para la revisión de hardware 2.

M.2 Configuración de I2C de la Raspberry Pi

I2C es un estándar comúnmente usado, diseñado para permitir la comunicación entre circuitos integrados. Ya que Raspberry Pi posee el protocolo I2C es posible conectar una gran variedad de integrados y módulos con salida I2C.

Antes de hacer uso de los puertos I2C deben de configurarse adecuadamente. Como primer paso debe de editarse el archivo con el siguiente comando en una terminal de shell.

```
sudo nano /etc/modules
```

luego agregar estas dos líneas al final del archivo

```
i2c-bcm2708
i2c-dev
```

Luego de editar el archivo debe de reiniciarse el sistema para que los cambios en los puertos físicos surjan efecto.

El bus de datos I2C permite la conexión de múltiples dispositivos a la Raspberry Pi, cada uno con una dirección única, que a menudo puede ser cambiada cambiando la configuración del módulo. Es muy necesario ver que dispositivos están conectados físicamente, así como también el determinar que el modulo conectado este trabajando apropiadamente.

Para poder hacer esto, debe de instalarse el programa i2c-tools desde la consola con el siguiente comando.

```
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
```

Dependiendo de la distribución, puede que se tenga un archivo en */etc/modprobe.d/raspi-blacklist.conf*

Si el archivo no existe, ya no se necesita hacer más nada, sin embargo y el archivo existe debe de comentarse las siguientes lineas.

```
blacklist spi-bcm2708
blacklist i2c-bcm2708
```

Colocando un # al inicio de cada línea

Para ello, debe de editarse el archivo con el siguiente comando:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Una vez hecho esto, puede usarse el siguiente comando y ver si todos los dispositivos están conectados (si se está utilizando el modelo B de 512MB de Raspberry Pi)

```
sudo i2cdetect -y 1
```

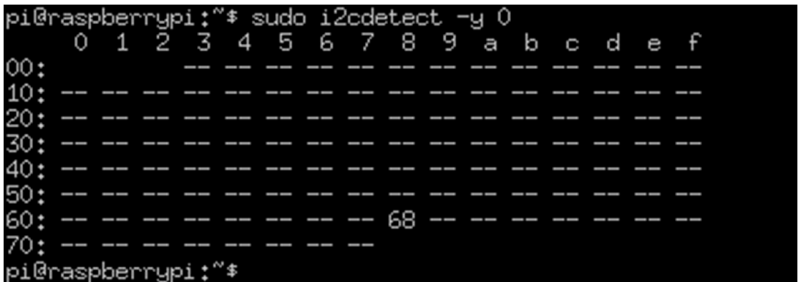


Figura 45 Direcciones de memoria usadas por I2C

La imagen anterior muestra la dirección de memoria que está siendo usada, 0x68 corresponde a la dirección del DS1307

Si se está utilizando la primera versión de Raspberry Pi (modelo B de 256MB de Raspberry Pi) se debe de cambiar el comando a:

```
sudo i2cdetect -y 0
```

Los diseñadores de Raspberry Pi cambiaron los puertos entre las versiones A y B. Vale la pena recordar que 512MB Raspberry Pi usa el puerto 1 de I2C. 256MB usa el puerto 0.

M.3 Establecer la hora al módulo RTC.

Una vez instalado el módulo RTC y verificado su funcionamiento con *i2cdetec*, se procede a configurar el módulo.

Primero se debe de cargar el modulo con el comando

```
sudo modprobe rtc-ds1307
```

Luego como superusuario el comando

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
```

(Si es la rev 1 de la Pi)

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

(Si es la rev 2 de la Pi)
Luego con el comando

```
sudo hwclock -r
```

Lee la hora del módulo DS1307. Si es la primera vez que el modulo es usado, reportara una fecha de Jan 1 2000, por lo que debe de configurarse la hora.

Para configurarlo se debe de tener la hora correcta de internet configurada en la Raspberry Pi. Automáticamente establecerá la hora correcta. Esto puede revisarse con el comando.

```
date
```

Para grabar la hora en el módulo RCT, se utiliza el siguiente comando.

```
sudo hwclock -w
```

Puede revisarse si los cambios fueron guardados nuevamente con el comando

```
sudo hwclock -r
```

A continuación se es necesario agregar el módulo kernel de la RTC en la lista de modulos /etc/modules, para que sea cargado cuando la maquina arranque.

Para agragarlo se utiliza el siguiente comando

```
sudo nano /etc/modules
```

Y agregar la linea

```
rtc-ds1307
```

Al final del archivo.

Luego para crear el dispositivo al iniciar el sistema, debe de editarse el archivo /etc/rc.local con el siguiente comando

```
sudo nano /etc/rc.local
```

Y luego agregar antes de la línea que contiene *exit 0* los siguientes comandos

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
```

*(*Si es la rev 1 de la Pi*)*

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

*(*Si es la rev 2 de la Pi*)*

```
sudo hwclock -s
```

(para ambas versiones)

Una vez que el sistema inicie, la hora deberá ser sincronizada con el módulo RTC.

Anexo N. Comparación de las plataformas como un servicio Google App Engine y Heroku.

Google App Engine y Heroku son sofisticadamente similares. Ambos proveen una solución de Plataforma como un servicio (PaaS). Ambos brindan un ambiente pre-desarrollado donde se pueden desarrollar aplicaciones. Esto significa que se puede tener corriendo una aplicación mucho más rápido, pero esto implica que debe de ser desarrollado únicamente dentro de la plataforma que se utilice.

A continuación se presenta una tabla donde se muestras las principales diferencias entre ambas plataformas.

Heroku: Ruby Ruby on Rails PostgreSQL, etc	**App Engine** Python webapp framework DataStore
Ventajas: Estándar SQL Desarrollo relativamente fácil Modelo de precio simple	**Ventajas:** Acceso al resto de servicios de Google Corren en la propia infraestructura de Google Ventajas para el uso gratuito(memcache, email, local static file storage)
Desventajas: No tan grande como Google. Hospedado en AWS	**Desventajas:** Poca flexibilidad para cambiarse de plataforma. No estándar SQL DB.

Tabla 17 Comparación entre Heroku y Google App Engine.

Bibliografía.

[1] Bonilla Perla, Juan José (2014) *Diseño, configuración y supervisión de la red de medidores de energía eléctrica del campus central de la Universidad de El Salvador.* Tesis EngD, Universidad de El Salvador. [Online] http://ri.ues.edu.sv/5584/

[2] Wireshark: Network Protocol Analyzer for Unix and Windows [Online] https://www.wireshark.org/

[3] Electroindustries Shark 100S. Manual De Instalación y Operación. Página 128 [Online]. http://www.electroind.com/pdf/sp/SP-100S-man.pdf

[4] Libmodbus, A Modbus library for Linux, Mac OS X, FreeBSD, QNX and Win32. Libmodbus - a fast and portable Modbus library. [Online]. http://libmodbus.org/

[5] Funcion modbus_get_float(). Libmodbus documentación [Online]. http://libmodbus.org/docs/v3.0.6/modbus_get_float.html

[6] Google App Engine Quotas [Online] https://developers.google.com/appengine/docs/quotas

[7] Google Chart Tools. API de JavaScript para visualización grafica [Online] https://developers.google.com/chart/?hl=es

[8] Google App Engine SDK for Python 2.7. [Online] https://developers.google.com/appengine/downloads?hl=es

[9] Annotated Time Line. Gráfico de tiempo interactivo y dinámico. [Online] https://developers.google.com/chart/interactive/docs/gallery/annotatedtimeline?hl=es

[10] Open arbitrary resources by URL [Online] https://docs.python.org/2/library/urllib.html