UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

INTERROGACIÓN DE MEDIDORES DE ENERGÍA UTILIZANDO EL PROTOCOLO DNP3

PRESENTADO POR:

JOSÉ ERNESTO DE PAZ ÁLVAREZ

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, OCTUBRE 2018

UNIVERSIDAD DE EL SALVADOR

RECTOR :

MSC. ROGER ARMANDO ARIAS ALVARADO

SECRETARIO GENERAL :

MSC. CRISTÓBAL HERNÁN RÍOS BENÍTEZ

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERIA ELÉCTRICA

DIRECTOR :

ING. ARMANDO MARTÍNEZ CALDERÓN

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

INTERROGACIÓN DE MEDIDORES DE ENERGÍA UTILIZANDO EL PROTOCOLO DNP3

Presentado por :

JOSÉ ERNESTO DE PAZ ÁLVAREZ

Trabajo de Graduación Aprobado por:

Docente Asesor :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

San Salvador, octubre 2018

Trabajo de Graduación Aprobado por:

Docente Asesor :

DR. CARLOS EUGENIO MARTÍNEZ CRUZ

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, martes 25 de septiembre de 2018, en el Aula de Posgrados de la Escuela de Ingeniería Eléctrica, a las 3:30 p.m. horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. Armando Martínez Calderón
   Director
   Firma

2. MSC. José Wilber Calderón Urrutia
   Secretario
   Firma

Y, con el Honorable Jurado de Evaluación integrado por las personas siguientes:

- DR. CARLOS EUGENIO MARTINEZ CRUZ
  (Docente Asesor)
  Firma

- MSC. JORGE ALBERTO ZETÍNO CHICAS
  Firma

- ING. WALTER LEOPOLDO ZELAYA CHICAS
  Firma

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

INTERROGACIÓN DE MEDIDORES DE ENERGÍA UTILIZANDO EL PROTOCOLO DNP3

A cargo del Bachiller:

- DE PAZ ALVAREZ JOSE ERNESTO

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final: 9.5

(NUEVE PUNTO CINCO

Agradecimientos

Jose Ernesto De Paz Álvarez

Agradezco profundamente a mis padres que han estado en mi apoyo desde mi nacimiento hasta este día. Mi padre es un hombre luchador que me ha animado y ha mantenido en alto mi autoestima desde que tengo memoria, hasta el día de la presentación de esta investigación y su entrega me hace saber que estará siempre con migo. Mi madre es una devota cristiana que ha dedicado sus oraciones por mi todo el tiempo y su apoyo hasta este día ha sido incondicional en todo aspecto. Estoy eternamente agradecido con mis padres y este logro les pertenece a ellos.

En cuestión de la realización de este trabajo, agradezco el apoyo a Omar Duque Debido a sus asesorías durante todo el proceso de la realización de esta investigación, y a que tuviese la amabilidad de poner a mi disposición parte del equipo necesario para la realización de la interrogación de los medidores, Luis Palacios y Samuel Renderos debido a la asesoría prestada al inicio de este proyecto.

Agradezco al Doctor Carlos Martínez por ser mi asesor en este trabajo de graduación y por su paciencia en los momentos difíciles de la realización del mismo. Así también agradezco a Adam Crain que es uno de los Gurús de OpenDNP3 quien en el momento culminante de este trabajo ayudo a solventar una duda fundamental que permitió el avance hacia el final de esta investigación.

Admiro grandemente a las personas que he mencionado en este apartado y de todo corazón les expreso mi más grande gratitud, debido a que sin su ayuda este trabajo de graduación no habría sido posible.

Gracias a todos.

Jose Ernesto De Paz Álvarez

Índice general

Lista de Figuras.

Capítulo 1: Introducción.

En la universidad se realiza un monitoreo constante del consumo de energía eléctrica debido a que este va en aumento cada año. A manera de contribuir a la observación de como ocurre dicho consumo en el tiempo, se realizó el proyecto de la instalación de medidores en algunas subestaciones de la universidad de las cuales se puede ahora conocer el comportamiento en el consumo energético [11].

En el año 2012 la Universidad de El Salvador implementó una red de medición en treinta de sus subestaciones, para lo cual se instalaron treinta medidores que monitorean diferentes tipos de variables eléctricas. Además a cada medidor se le instaló un router inalámbrico tipo WiFi. Todos los router forman parte de una red malla o mesh [1]. Hasta la fecha se cuenta con medidores en 26 subestaciones y todas ellas son interrogadas mediante el protocolo Modbus TCP. Sin embargo, los medidores tienen la capacidad de comunicarse por otros protocolos de comunicación, los cuales al explorarse pueden ampliar, la cartera de fabricantes a los que acudir para adquirir medidores de energía, en caso que en el futuro desee hacerse una ampliación de la red.

El trabajo se divide en 7 partes estructuradas en 4 capítulos. En la primera parte se habla de generalidades del tema. En la segunda se conocen las características físicas de los medidores y la configuración manual de los mismos. En la tercera parte, se detallan las pruebas de comunicación realizadas con el software original de la empresa *Electro industries* por medio del protocolo Modbus RTU. En la cuarta parte, se detallan códigos en base a la biblioteca LibModbus en lenguaje C para Linux, con los cuales pueda realizarse la interrogación mediante los protocolos Modbus RTU y Modbus TCP/IP. En la quinta parte, se detallan pruebas de comunicación con el simulador OpenDNP3. En la sexta parte se realizan pruebas con programas específicos para el protocolo DNP3 RTU utilizando la biblioteca OpenDNP3 y la realización de algunos ejemplos. En la séptima parte, se hace un escaneo del puerto serial con el software Wireshark para observar la comunicación mediante DNP3. Estas 7 partes, se encuentran mejor detalladas en la sección 1.5.

1.1 Interés de la investigación

Actualmente en la universidad de El Salvador, se cuenta con subestaciones que están siendo monitoreadas por medidores de energía SHARK 100S y SHARK 200 y SHARK 200S. Los medidores de energía se han comunicado hasta hoy por medio del protocolo modbus TCP y sobre esa base, se ha montado y desarrollado la red de medidores existente en la universidad. En este trabajo se ampliará la cartera de protocolos de comunicación explorando la comunicación por medio del protocolo DNP3 RTU.

1.2 Antecedentes

Los medidores de energía fueron instalados en la universidad, en el año 2012 en diferentes subestaciones de la universidad, para los cuales se decidió implementar la interrogación por medio del protocolo Modbus TCP/IP [1]. Según se observa en la Figura 1, los datos son transmitidos al router MP que está conectado directamente al medidor, mediante la conexión ethernet. Posteriormente al ocurrir el enlace, mediante el protocolo B.A.T.M.A.N los datos son enviados desde el router conectado al medidor, hasta idealmente el router conectado al servidor (En la realidad la información puede dar saltos entre varios routers Mesh 01 hasta llegar al servidor) [15]. El servidor es una raspberry pi 2 modelo B que posteriormente se encarga del procesamiento de los datos recibidos. El esquema de la red de medidores, se muestra en la Figura 1.

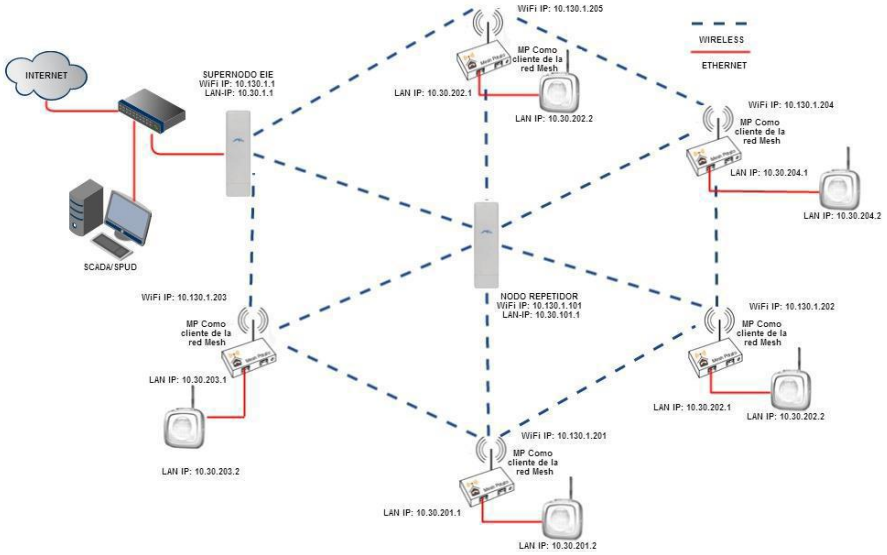

Figura 1. Esquema de la red de medidores [1].

La Figura 1, muestra el diagrama de la red malla implementada en la universidad.
Cabe destacar, que los medidores ya mencionados, tienen la capacidad de comunicarse por protocolo Modbus utilizando el puerto Ethernet (Modbus TCP/IP), el puerto RS485 (protocolos Modbus RTU y DNP3 RTU), y posee también un puerto IrDA con el cual se puede interrogar de manera infrarroja.

1.3 Motivación para realizar el proyecto

La razón principal de esta investigación es ampliar la gama de protocolos por las cuales puedan ser interrogados los medidores de energía. De esta manera la red ensamblada con medidores de la empresa *Electro industries*, podría ser ampliada con medidores de otros fabricantes que puedan comunicarse por medio de los protocolos a explorar en este trabajo (Modbus RTU y DNP3 RTU). De esta manera se puede llegar a ampliar la oferta de fabricantes de medidores más allá de los ofrecidos por la empresa *Electro industries.* Se podrá comparar diferentes fabricantes y valorar cuestiones como el precio y las características. Con ello se podrá tomar mejores decisiones al momento de elegir hacer una ampliación de la red.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un mecanismo de interrogación sobre una Raspberry utilizando el protocolo DNP3. Para interrogar los medidores SHARK 100, SHARK 200 Y SHARK 200S, por medio del puerto RS485.

1.4.2 Objetivos específicos

- Obtener comunicación entre un dispositivo maestro y el medidor. La interrogación se realizará mediante el software del fabricante, *Electro industries,* para protocolo Modbus RTU y mediante el simulador de OpenDNP3 para comunicación por DNP3.

- Utilizar el módulo USB-RS485 y comprobar que éste produce resultados satisfactorios en todas las pruebas.

- Realizar códigos en lenguaje C con ayuda de la biblioteca LibModbus e interrogar el medidor tanto por Modbus TCP como por Modbus RTU.

- Realizar códigos en lenguaje C++ en base a la biblioteca OpenDNP3 e interrogar el medidor mediante DNP3 RTU.

- Realizar compilación común de Linux para códigos en C con librería LibModbus y compilación con CMake para códigos de la biblioteca Open DNP3 en C++.

- Ingresar datos en una base de datos MySQL.

- Hacer monitoreo del puerto serie con el software Wireshark.

- Dotar a los medidores SHARK 100 de la capacidad de almacenamiento de datos y ampliar dicha capacidad a los medidores SHARK 200.

- Ampliar el abanico de fabricantes posibles a los cuales se pueda acceder.

1.5 Organización

El trabajo se divide en 7 partes estructuradas en 4 capítulos. La primera parte está contenida en el primer capítulo. En ella se habla acerca de las generalidades del tema. El segundo capítulo, contiene 3 de las partes subsiguientes. En la segunda parte, se muestran características físicas del medidor entre ellas los botones y sus respectivas funciones, los puertos disponibles, y el convertidor USB-RS485 utilizado para todas las pruebas de comunicación. También, se detalla como configurar de manera manual los medidores por medio de las teclas cursoras físicas del equipo. Se explica cómo variar los parámetros básicos como Baud Rate, dirección, Protocolo de comunicación, entre otros parámetros. En la tercera parte, mediante el módulo convertidor USB-RS485, ya mencionado, se detallan las pruebas de comunicación realizadas con el software original de la empresa *Electro industries* por medio del protocolo Modbus RTU. Se consigue mostrar la cantidad de datos que este software ofrece y lo que se puede hacer con ellos. En la parte 4, de este mismo capítulo, se detallan códigos en base a la biblioteca LibModbus en lenguaje C para Linux, con los cuales se realiza la interrogación mediante los protocolos Modbus RTU y Modbus TCP. El tercer capítulo, contiene las 3 partes finales. En la parte 5 se detallan pruebas de comunicación con el simulador de OpenDNP3. Se corrobora que la interrogación por medio del protocolo DNP3 en efecto puede realizarse. En la parte 6, se implementan programas específicos para el protocolo DNP3 RTU, utilizando la biblioteca OpenDNP3. Se especifican los parámetros a variar en los mismos para poder interrogar medidores que funcionen con el protocolo DNP3 TCP. También, se realizan ejemplos entre los cuales se hace una interrogación básica, conversión de los datos recibidos para mostrar los valores analógicos en la terminal de linux e impresión en archivos de texto. Inclusive, envío de datos a tablas de bases de datos en MySQL. En la parte 7, se muestra una tabla en la que se muestra el resultado de un escaneo al puerto serie al momento de ocurrir la comunicación DNP3 utilizando Wireshark.

Capítulo 2: Interrogación preliminar de los medidores.

2.1 Puertos y protocolos de comunicación existentes en los medidores SHARK.

Los medidores de energía tienen dos puertos físicos de conexión para la comunicación. Por una parte, se tiene el puerto ethernet que funciona únicamente para comunicación Modbus TCP. Por otra parte, está el puerto serie RS485, con el cual es posible interrogar a los medidores utilizando los protocolos Modbus RTU, Modbus ASCII y DNP3 RTU. Para estos medidores no se encuentra disponible la comunicación DNP3 por TCP, por lo cual para poder interrogar por DNP3 debemos utilizar el puerto serie. Sin embargo, en la sección 3.6 se muestran los cambios que se deben realizar al código para poder comunicar medidores de otros fabricantes mediante DNP3 TCP.

2.2 Configuración manual de parámetros de los medidores.

Los medidores SHARK tienen una forma de presentar los datos en el mismo medidor con displays de 7 segmentos como puede verse la Figura 2. La caratula de los medidores poseen dos teclas cursoras y dos teclas que permiten la configuración manual. Esenciales como el Baud Rate, protocolo de comunicación y la dirección especifica del medidor. Esos botones son “Menu” y “Enter” [9].

Como primer paso debe encenderse el medidor conectándolo a la toma de energía. Al encenderse y haber esperado un breve periodo de tiempo, el medidor se muestra como se observa en la Figura 2(a).

En ese momento el medidor está listo para configurarse manualmente, en caso de querer variar sus parámetros por defecto. Entre los más importantes están la dirección RTU, el protocolo de comunicación y la velocidad de comunicación serie (Baud Rate).

Para variar dichos parámetros se presiona el botón “Menu” una vez, el medidor aparecerá como en la Figura 2(b). Luego de presionar “Menu”, para poder desplazarse entre las diferentes opciones se debe presionar la tecla “Enter” y comenzaran a variar los datos en pantalla.

Si, por ejemplo, lo que se desea es variar la velocidad de comunicación serie, luego de haber presionado “Menu”, se presiona “Enter” las veces que sea necesario hasta llegar a esa opción, como se muestra en la figura 2(d). Esta velocidad se puede cambiar a diferentes valores: 9600, 38400, 57600 y 115200 Baudios.

De la misma manera puede variarse el protocolo de comunicación entre las tres opciones

que el medidor ofrece: Modbus RTU, Modbus ASCII y DNP3. En la Figura 2(e) se muestra la caratula del medidor cuando se ha seleccionado el protocolo DNP3.

Otro ejemplo seria configurar la dirección serie del medidor. Esta dirección varía entre 0 y 999 en la caratula del medidor aunque teóricamente se puede contar hasta 65536 direcciones ($2^{16} = 65536$). El número se puede variar con las teclas cursoras. Con la que apunta hacia la derecha, puede cambiarse el digito que se desea configurar, y con la que apunta hacia abajo se cambia el número del display entre 0 y 9. En la Figura 2(c) se observa la dirección 002 programada al dispositivo.

Es importante tener presente que después de haber realizado cualquier cambio éstos deben de ser guardados. El procedimiento para guardar los cambios requiere de dos pasos. Primero se presiona dos veces el botón “Menu”, véase Figura 2F. Luego, se presiona “Enter” para guardar los cambios y se observará que de manera automática el medidor se reinicia. Se puede verificar que el medidor se ha reiniciado y ha guardado los cambios de manera exitosa si se observa lo que se muestra en la Figuras 2(g), 2(h) y 2(i).

Posteriormente solo se debe esperar un breve periodo de tiempo hasta que vuelva a mostrarse la Figura 2(a). El medidor puede utilizarse ya con la nueva configuración.

Figura 2. Configuración de medidores. (a)Pantalla inicial de los medidores luego de encenderse. (b) medidor preparado para recibir posibles configuraciones. (c) Variación de la dirección del medidor. (d) Variación del Baud Rate del medidor. (e) Variación del protocolo de comunicación del medidor. (f) Medidor listo para resetearse y guardar los cambios aplicados. (g)(h)(i) Imágenes que muestran la forma de reinicio del medidor.

2.3 Comunicación mediante USB-RS485 con *Comunicator Ext*.

Como prueba preliminar de comunicación se utilizó el conector USB- RS485 que se muestra en la Figura 3. La parte USB se conecta a la computadora y la parte RS485 al medidor de energía.

Figura 3. Convertidor USB-RS485.

El medidor debe configurarse como se muestra en la Figura 4. Es decir, el *Jumper* debe estar uniendo los dos pines inferiores como lo muestra el recuadro 3.

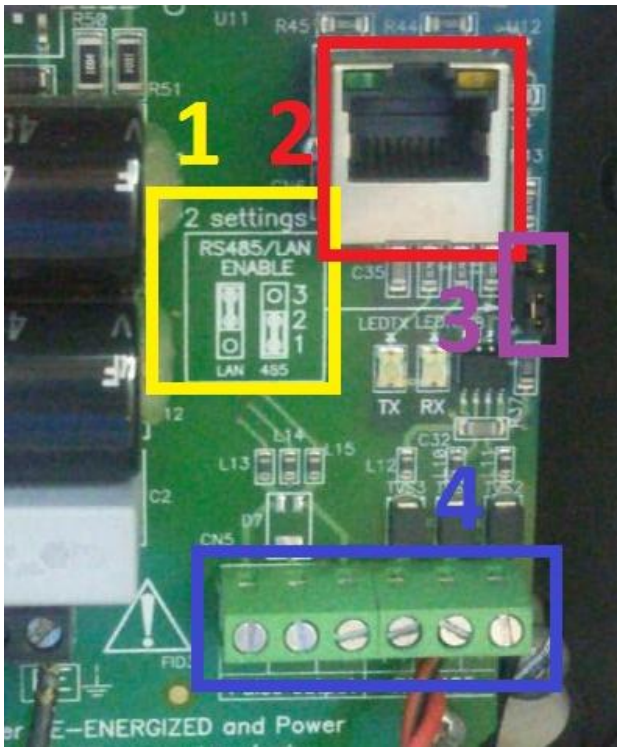

Figura 4. Configuración del *Jumper* para pasar de TCP/IP a comunicación Serie.

El recuadro número 1 de la Figura 4 (mostrado en amarillo) muestra cómo debe configurarse el medidor en el caso de querer utilizar cualquiera de los puertos para interrogar al medidor. El *Jumper* posee tres pines. En caso de interconectar los dos pines de arriba, se da acceso a la conexión LAN (Ethernet), y conectado el *Jumper* entre los dos pines de abajo, se da acceso al puerto serie RS485.

El recuadro número 2 de la Figura 4 (mostrado en rojo) es el puerto Ethernet del medidor. El cual puede conectarse de manera directa a una computadora, una Raspberry pi o un Router MP01.

El recuadro número 3 de la Figura 4 (mostrado en purpura). Muestra el *Jumper* físicamente en el cual están conectados los dos pines de abajo. Lo que implica que el medidor está configurado para poder ser interrogado mediante el puerto serie RS485.

El recuadro número 4 de la Figura 4 muestra físicamente el puerto serie RS485.De éste, se desprenden 2 cables que van hacia el conector USB-RS485 y dan paso a la comunicación serial. Luego de la conexión correcta del *Jumper* y el conector USB-RS485 se debe configurar el medidor como se especificó en la sección 2.2. Se debe tener en cuenta que, en el lado de la computadora, debe instalarse un driver para que el dispositivo USB-RS485 sea reconocido por la misma.

Una vez configurado el Jumper correctamente e instalado el driver del conector USB-RS485, se procede a instalar el software *Comunicator Ext* en la computadora. Este software está disponible en una versión reducida (lite) de forma gratuita y solo corre en sistema operativo Windows [10].

Una vez instalado el software se procede a establecer una conexión con el medidor. Para lo cual se abre el programa *Comunicator Ext,* como se muestra en la Figura 5. Para realizar una conexión con un medidor, debe elegirse la pestaña "connect", véase la Figura 6.

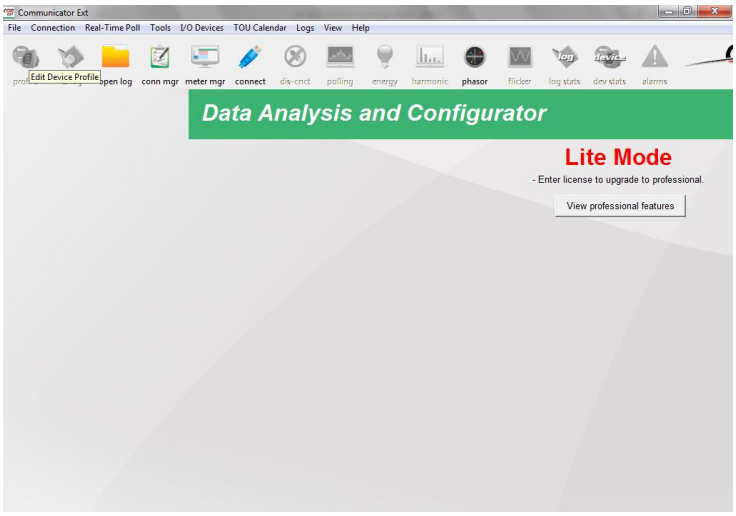

Figura 5. Interfaz principal de Software *Comunicator Ext*.

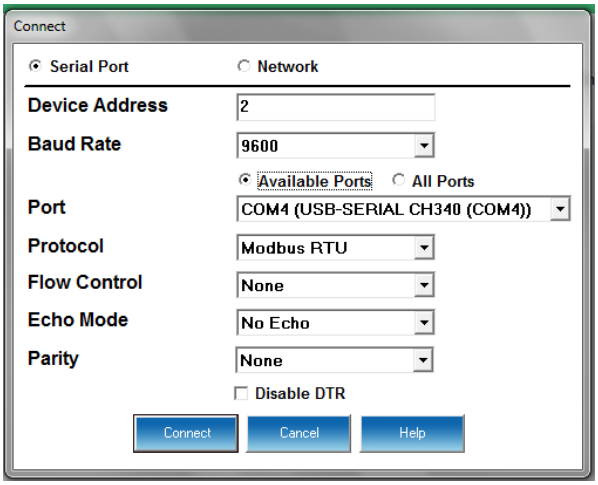

Figura 6. Pestaña "Connect", donde se setean los parámetros colocados en el medidor.

El recuadro “connect” de la Figura 6 debe rellenarse con los datos previamente configurados al medidor. Se selecciona el boton “conect” y aparece el perfil inicial del medidor de energía como se muestra en la Figura 7.

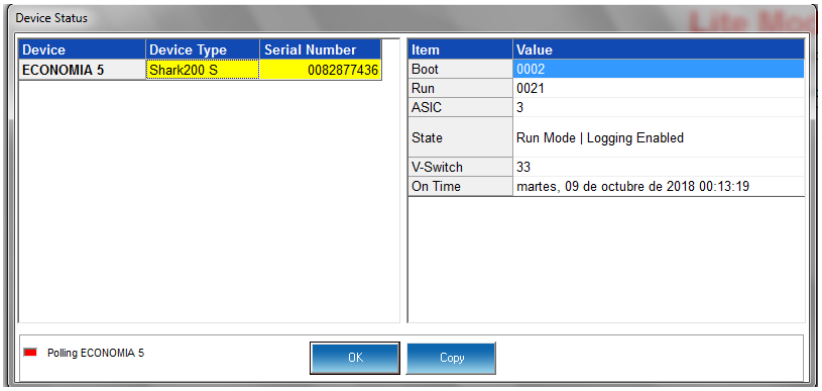

Figura 7. Correcto resultado de comunicación con un medidor SHARK 200S.

Con esto se ha establecido la comunicación mediante el protocolo Modbus RTU. Luego de establecida la conexión, puede buscarse en la interfaz dos de las opciones más importantes “Polling” y “Energy”. La opción “Polling” hace una interrogación de muchos datos los cuales están cambiando en tiempo real. Entre ellos, voltajes, corrientes, potencia real, potencia aparente, potencia reactiva, etc. La opción “Energy” muestra datos de potencia total, de potencia por fases, de energía consumida, etc.

En la versión lite del programa solo se puede interrogar mediante las variaciones mencionadas anteriormente del protocolo Modbus (ASCII, TCP y RTU). La versión lite del *Comunicator Ext* no permite la comunicación mediante el protocolo DNP3. Para poder interrogar al medidor mediante DNP3 con el software del fabricante debe obtenerse la versión profesional de pago.

Aun si se realizara el pago, para tener a la mano ese servicio, no es posible procesar los datos enviándolos a una base de datos para realizar estadísticas. Por lo cual en las secciones posteriores, la interrogación se hace mediante códigos escritos en c y c++ para Modbus y DNP3 respectivamente, para tener mayor control sobre los datos recibidos.

2.4 Interrogación de medidores mediante Modbus RTU.

2.4.1 Protocolo Modbus.

Modbus es un protocolo de comunicaciones, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Debido a que este protocolo fue público, de fácil uso y que requiere poco desarrollo (maneja bloques de datos sin suponer restricciones) se convirtió

en un protocolo de comunicaciones estándar en la industria. Es el protocolo de mayor disponibilidad para la conexión de dispositivos electrónicos industriales. El protocolo Modbus permite el control de una red de dispositivos, por ejemplo, un equipo de medición temperatura y humedad puede comunicar los resultados a una PC. Modbus también se usa para la conexión de un PC de supervisión con una unidad remota (RTU) en sistemas de supervisión de adquisición de datos (SCADA). Existen versiones del protocolo Modbus para puerto serial y Ethernet (Modbus/TCP) [1].

2.4.2 Librería LibModbus.

Libmodbus es una librería de código abierto para el envío y recepción de datos utilizando el protocolo de comunicación Modbus. Esta librería está escrita en C y soporta las variantes RTU (serial) y TCP (Ethernet). Licenciada bajo LGLP 2.1+, eso implica que se puede conocer el código fuente, estudiarlo y modificarlo para adaptarlo a nuestra necesidad como se ha hecho en este trabajo. La librería es una implementación de alto nivel del protocolo Modbus tanto serial, TCP/IPv4 y TCP/PI IPv4 e IPv6. Tiene funciones para establecer comunicación Modbus TCP/IP y manejo de datos que utilizamos en el programa de adquisición de datos que desarrollamos en lenguaje C para comunicarnos e interrogar los medidores SHARK 100S, SHARK 200 y SHARK 200S [1].

2.4.3 Interrogación.

Luego del establecimiento de comunicación y establecida la conexión principal USB - RS485, se puede proceder a armar un programa con la ayuda de la librería Libmodbus. Se debe especificar los datos del puerto serie para que la comunicación sea Modbus RTU. El código que se utilizó para la interrogación se muestra en la figura 8.

```
1. #include <stdio.h>
2. #include <errno.h>
3. #include <modbus/modbus.h>
4. #include <modbus-rtu.h>
5. int main()
6. {
7. modbus_t *ctx;//Crea puntero
8. uint16_t tab_reg[64]; //Crea un arreglo unsigned long int para
   colocar los datos
9. uint16_t toFloat[2];
10.     uint16_t toInt[2];
11.     int rc;
12.     int i;
13.     float Voltaje_A=0;
```

```
14.      /* Establecer un contexto Modbus */
15.      ctx = modbus_new_rtu("/dev/ttyUSB0", 57600, 'N', 8,1);
16.      if (ctx == NULL) {
17.         fprintf(stderr, "No pudo ubicar el contexto libmodbus\n");
18.         return -1;
19.      }
20.      /* Establece la conexion Modbus */
21.      modbus_set_slave (ctx, 001);
22.      if (modbus_connect(ctx) == -1) {
23.         fprintf(stderr, "La conexion fallo:
   %s\n", modbus_strerror(errno));
24.         modbus_free(ctx);
25.         return -1;
26.      }
27.      /* Lee 10 registros desde la dirección 0 */
28.      rc=modbus_read_registers(ctx, 999, 2, tab_reg); //Lee
   registros
29.      if (rc == -1) {
30.         fprintf(stderr, "Error en la adquisicion de los datos
   %s\n", modbus_strerror(errno));
31.         return -1;
32.      }
33.      for (i=0; i < rc; i++) {
34.         printf("reg[%d]=%d (0x%X)\n", i, tab_reg[i], tab_reg[i]);
35.      }
36.      toFloat[0]=tab_reg[1];
37.      toFloat[1]=tab_reg[0];
38.      Voltaje_A= modbus_get_float(toFloat);
39.      printf("Voltaje Fase A: %f\n",Voltaje_A;
40.      /* Cierra la conexion Modbus */
41.      modbus_close(ctx);
42.      modbus_free(ctx);
43.      return(0);
44.      }
```

Figura 8. Código completo en lenguaje C para interrogación por Modbus RTU.

En las primeras 4 líneas del código se declaran las librerías que se utilizarán de la biblioteca LibModbus. En la línea 5 se declara la función principal. En la línea 7 se declara el puntero que manejara todo el mapa Modbus. Las líneas 8 - 13 realizan la declaración de variables necesarias. Entre ellas una flotante que se encargará de mostrar el valor analógico de la interrogación, variables enteras, y dos tablas de 16 bits que servirán para almacenar los datos que vienen del medidor, antes y posteriores a su conversión a valor flotante (los datos vienen cifrados en los registros al realizarse la interrogación).

Las líneas 15 - 19 muestran la forma de configurar el puerto serie. Los parámetros más importantes son: El nombre del puerto y la velocidad (bits por segundo). En caso de ocurrir un error en la comunicación, se imprimirá el mensaje “no se pudo ubicar el contexto LibModbus”.

Las líneas 21 - 26, especifican al medidor cual será la estación remota. Asignándole una dirección la cual debe ser igual a la que se le programe al medidor. De lo contrario no habrá comunicación y de existir un error en esa porción del programa se imprimirá el error “La conexión falló”.

Las líneas 28 – 32 se encargan de la adquisición de los datos del mapa Modbus y de su respectivo almacenamiento en la tabla que se asigna. Si por alguna razón no puede realizarse la lectura el programa responderá en ese momento “Error en la adquisición de datos”. En el caso que la lectura de datos tenga éxito, los datos pasaran a mostrarse con las instrucciones que se muestran entre las líneas 32 - 35. En ese caso se mostrarán los datos que devuelve el medidor como registros de 16 bits [1]. Las líneas 3 - 40 realizan la conversión de los registros de 16 bits.

Por último, las líneas 43 - 46 cierran la comunicación Modbus.

En este caso, es posible realizar una interrogación mediante el protocolo Modbus RTU. Es muy importante destacar que para que la comunicación pueda realizarse, el medidor debe de tener el *Jumper* según se muestra en la Figura 4.

En caso de quererse realizar comunicación mediante el puerto Ethernet, por medio del protocolo Modbus TCP, debe colocarse el *Jumper* de la Figura 4 entre los dos pines de arriba. Las líneas 15- 19 del código de la Figura 8 se sustituyen como se muestra en la Figura 9.

```
15.      ctx = modbus_new_tcp("10.0.0.1", 502);
16.      if (ctx == NULL) {
17.         fprintf(stderr, "No pudo ubicar el contexto libmodbus\n");
18.         return -1;
19. }
```

Figura 9.Cambio en el código para obtener comunicación mediante Modbus TCP.

Donde “10.0.0.1” es la IP del destino (medidor) y el puerto mediante el cual se dará la comunicación es el 502.

Para consultar el mapa Modbus, como por ejemplo voltajes de fase, voltajes de línea, parámetros del fabricante, nombre del medidor, número de serie, etc. Se deben configurar las siguientes líneas como se muestra en la Figura 10.

```
28.      rc=modbus_read_registers(ctx, 0, 10, tab_reg);
29.      if (rc == -1) {
30.         fprintf(stderr, "Error en la adquisicion de los datos
   %s\n", modbus_strerror(errno));
31.         return -1;
32. }
```

Figura 10. Lectura de registros del mapa Modbus.

La Figura 10 muestra el caso donde se realiza una lectura desde la dirección 0 del mapa Modbus y se leen 10 registros. En caso de querer consultar otros parámetros del mapa Modbus, puede consultarse en el manual del fabricante en [9].

2.4.3.1 Compilación de librería LibModbus y permisos para puerto Serie.

Para realizar la compilación del código se escribe lo siguiente en la línea de comandos de Linux, teniendo en cuenta que se debe de indicar el nombre real del archivo y el nombre que se le desee dar al ejecutable que aparecerá.

```
root@jose-Latitude-D510:/home/jose# gcc nombre_del_archivo.c –o nombre_para_el_ejecutable
'pkg-config-libs –-cflags libmodbus'
```

Luego de ya compilado el programa, y creado ya el ejecutable, al conectar el USB-RS485, el conector es reconocido como "USB0" y se le debe dar permisos al puerto serial para poderlo utilizar. En la terminal de Linux se escribe la siguiente línea.

```
root@jose-Latitude-D510:/home/jose# chmod 666 /dev/tty/USB0
```

Si la línea de los permisos del puerto no se escribe, el puerto no se abrirá, y al ser ejecutado el programa este nos notificara errores de conexión y la comunicación no ocurrira.

Luego de la compilación se genera el ejecutable, y la línea a escribir en la línea de comandos, para correr el ejecutable es la siguiente.

```
root@jose-Latitude-D510:/home/jose# ./nombre_del_ejecutable
```

En este caso se realizó la interrogación del voltaje en la fase A del medidor. El nombre del codigo es "Voltaje_A.c" y en el, se interroga la dirección del mapa Modbus correspondiente al voltaje en la fase A en el mapa Modbus. La Figura 11 muestra la interrogación. Y en ella aparece primero la petición de permisos para el uso del puerto serial, posteriormente la compilación del programa, y por último se corre el programa que en este caso al ejecutable generado se le llama Voltaje_A mostrando la tensión en la realización de 5 interrogaciones.

```
root@jose-Latitude-D510:/home/jose/Escritorio# chmod 666 /dev/ttyUSB0
root@jose-Latitude-D510:/home/jose/Escritorio# gcc Voltaje_A.c -o Voltaje_A `pkg-config --libs --cflags libmodbus`
root@jose-Latitude-D510:/home/jose/Escritorio# ./Voltaje_A
reg[0]=0x42EE
reg[1]=0xAC85
Voltaje Fase A: 119.336952
root@jose-Latitude-D510:/home/jose/Escritorio# ./Voltaje_A
reg[0]=0x42EE
reg[1]=0xA317
Voltaje Fase A: 119.318535
root@jose-Latitude-D510:/home/jose/Escritorio# ./Voltaje_A
reg[0]=0x42EE
reg[1]=0x9DB6
Voltaje Fase A: 119.308029
root@jose-Latitude-D510:/home/jose/Escritorio# ./Voltaje_A
reg[0]=0x42EE
reg[1]=0x8F3A
Voltaje Fase A: 119.279739
root@jose-Latitude-D510:/home/jose/Escritorio# ./Voltaje_A
reg[0]=0x42EE
reg[1]=0x8E79
Voltaje Fase A: 119.278267
root@jose-Latitude-D510:/home/jose/Escritorio#
```

Figura 11. Obtención de diferentes lecturas de datos mediante MODBUS RTU.

En este caso se ha interrogado el voltaje de la fase A que es la que provee de energía al medidor para su funcionamiento. Pero pueden interrogarse otras variables como por ejemplo: Corriente, voltaje, energía, potencias entre otros [9].

Es importante aclarar que la interrogación de datos se hace de manera diferente entre los protocolos Modbus y DNP3. Esto se debe principalmente a que los protocolos son distintos y por lo cual los datos se manejan de manera distinta.

En el capítulo siguiente se detallan datos fundamentales sobre el protocolo DNP3. También, se realizan pruebas con el simulador de OpenDNP3. Similarmente se realizan pruebas con el software *Comunicator Ext* que se mencionó en este capítulo. También se muestran distintos ejemplos de programas con la biblioteca OpenDNP3 de C++ estructurados específicamente para la interrogación de los medidores. Por último se analiza el comportamiento del puerto serie con el analizador de protocolos WireShark.

Capítulo 3: Interrogación de medidores mediante DNP3.

3.1 Protocolo DNP3.

3.1.1 Arquitectura del protocolo DNP3.

El protocolo de comunicaciones DNP3 define dos capas (Capa de Aplicación y Capa de enlace de Datos) y una subcapa (Función de Transporte), como muestra la Figura 12.
En la Figura 13 se muestra la arquitectura de protocolos DNP3 en una interconexión de una estación maestra (computadora) con una estación remota (medidor). [14]

A continuación, se describe la secuencia de eventos para enviar una orden de la estación maestra a la estación remota:

1. En la estación maestra se recibe la señalización de un determinado Punto a través de la Capa de Usuario y se inicia un proceso de solicitud en la Capa de Aplicación.
2. La Capa de Aplicación fragmenta la información recibida y pasa los fragmentos obtenidos a la Función de Transporte.
3. La Función de Transporte toma los fragmentos y los segmenta para pasarlos a la Capa de Enlace de Datos.
4. La Capa de Enlace de Datos obtiene los segmentos, genera la trama DNP3 y la envía a través del medio físico existente.
5. La Capa de Enlace de Datos de la estación remota recibe la trama del medio físico, elimina la cabecera de la trama obtenida y pasa la información a la Función de Transporte.
6. La Función de Transporte une los segmentos y pasa a la Capa de Aplicación.
7. La Capa de Aplicación desfragmenta la información recibida e indica la solicitud a la Capa de Usuario de la estación remota correspondiente.
8. Para dar respuesta a la solicitud de la estación maestra, la estación remota realiza el proceso inverso y espera por la confirmación del envío de los datos.

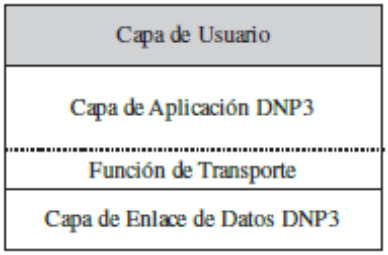

Figura 12. Capas en protocolo DNP3 [14].

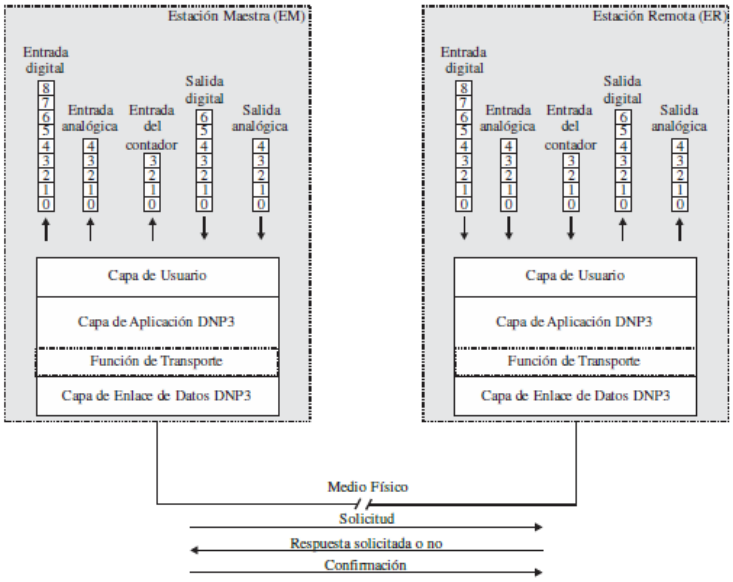

Figura 13. Transferencia de datos entre Maestro y Estación Remota de protocolo DNP3 [14].

En la Figura 14 se muestra cómo se construye la trama DNP a partir de un mensaje de la Capa de Aplicación. Dependiendo de su longitud, la Capa de Aplicación divide el mensaje en diferentes fragmentos y a cada fragmento le añade una cabecera de la Capa de Aplicación (AH, Application Header). La Función de Transporte recibe los fragmentos y los divide en segmentos (segment), a los cuales les añade una cabecera de la Función de Transporte (TH, Transport Header). Por último, la Capa de Enlace de Datos toma cada segmento, le añade la cabecera de la Capa de Enlace de Datos (LH, Link Header) y calcula el CRC para conformar la trama DNP3 [14].

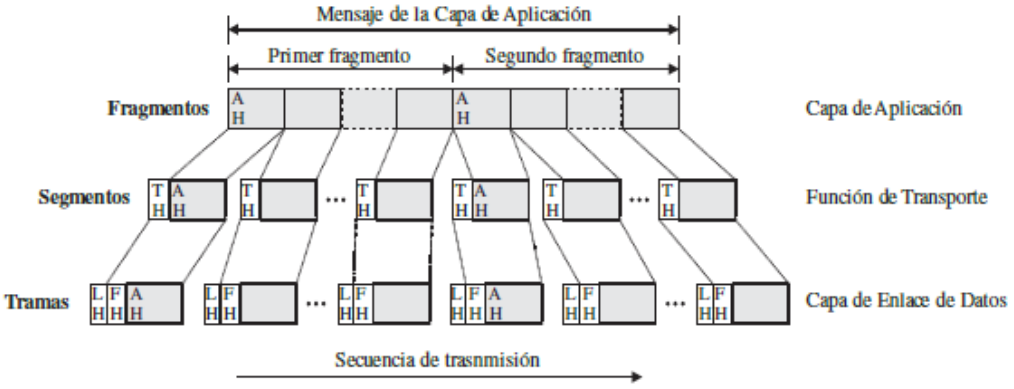

Figura 14. División del mensaje en las distintas capas del protocolo DNP3 [14].

La Figura 15 muestra las secuencias de envío y recepción de una solicitud, así como de su respuesta. Es importante ver como al pasar el mensaje de la capa de enlace del maestro, al mundo físico, y llegar a la capa de enlace de la estación remota y viceversa, se envían “ACKs” de confirmación que el dato ha llegado correctamente a su destino. La

Figura 16 ilustra la forma en que se envía una respuesta no solicitada y su confirmación.

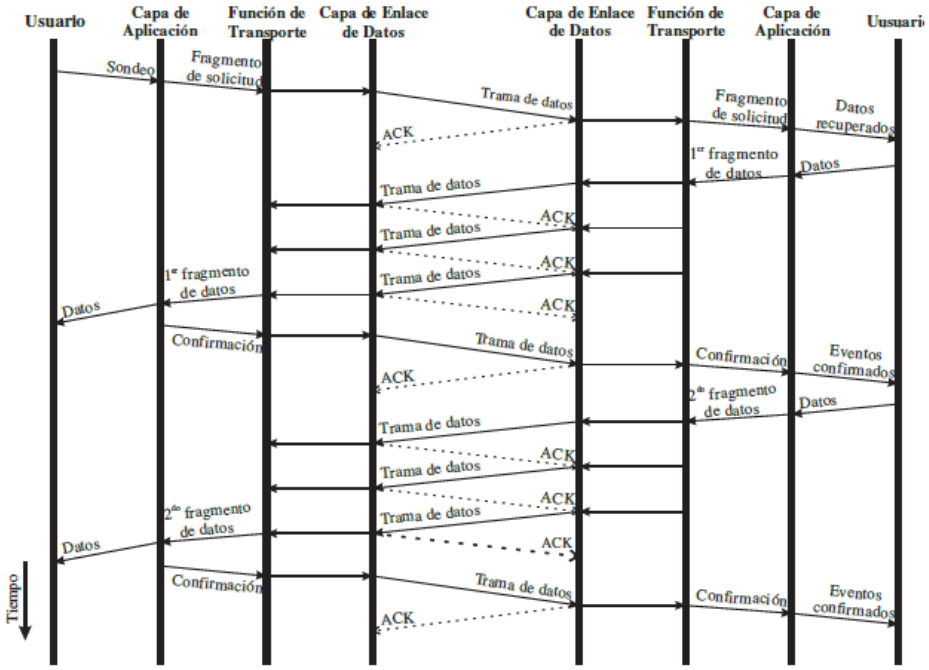

Figura 15. Trama de envío de una petición, confirmaciones y respuestas en el protocolo DNP3 [14].

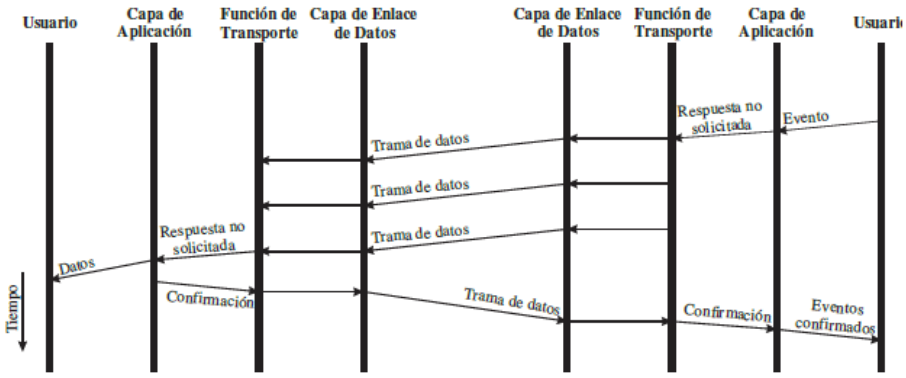

Figura 16. Respuesta no solicitada por parte de la estación remota [14].

3.2 Capas en DNP3.

3.2.1 Capa de aplicación.

El modelo de referencia OSI define a la Capa de Aplicación como la interfaz entre el software del usuario (Capa de Usuario) y las capas inferiores del protocolo (Figura 17), así mismo proporciona funciones estandarizadas, formatos de datos y procedimientos para la transmisión eficiente de datos, atributos y órdenes de control [12].

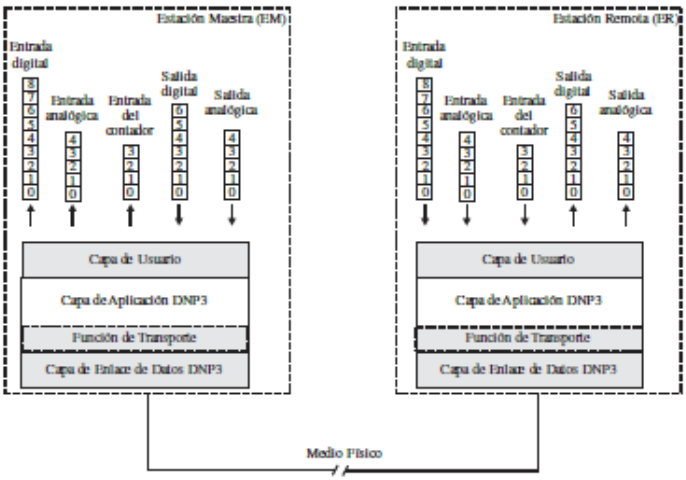

Figura 17.Capa de aplicación [12].

3.2.2 Subcapa de transporte.

El nivel de transporte es el encargado de permitir mensajes únicos estructurados tanto en múltiples tramas como en múltiples fragmentos.

El nivel de aplicación pasa los fragmentos al nivel de transporte, y este se encarga de trocearlos y agregarles al principio la cabecera de transporte, la cual ocupa un byte y contiene el número de secuencia que identifica el segmento dentro del fragmento. El tamaño de los fragmentos ha de ser tal, que una vez agregadas las cabeceras del nivel de enlace (diez bytes) y los correspondientes CRCs, el tamaño total no exceda los 292 bytes máximos permitidos para una trama.

En este caso, el nivel de transporte se encarga de recomponer los fragmentos del nivel de aplicación a partir de los segmentos que le proporciona el nivel de enlace. Para ello, recurre a las cabeceras de transporte y al número de secuencia que identifica la posición de cada segmento dentro del fragmento [12].

3.2.3 Capa de enlace.

Los mensajes DNP3 a nivel de enlace se encuentran en bloques de no más de 292 bytes denominados tramas. El formato de trama es similar al FT3, si bien presenta ciertas diferencias. Una trama DNP3 consta de dos bloques bien diferenciados. La cabecera y los datos:

3.2.3.1 Cabecera.

Son los diez primeros bytes de la trama, y está constituida por los siguientes campos:

1. bytes de inicio (start bytes), cuyo valor es fijo. 0x05 (valor en hexadecimal) para el primero y 0x64 para el segundo.
2. 1 byte con el tamaño de la trama. Este valor no tiene en cuenta ni la cabecera, ni los CRC.
3. 1 byte con el código de control, que permite fijar los servicios del nivel de enlace, el sentido del flujo, etc.
4. 2 bytes con la dirección de destino, codificada en big-endian.

5. 2 bytes con la dirección de origen, codificada en big-endian.
6. 2 bytes de CRC.

3.2.3.2 Datos.

Cada 16 bytes de datos, así como al final de la trama, se encontrarán 2 bytes de **CRC**.

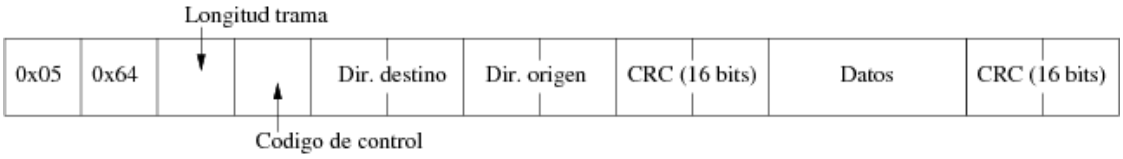

Figura 18. Estructura general de las tramas de bits en la capa de enlace en DNP3 [12].

El nivel de enlace en DNP es balanceado, de modo que tanto la estación controladora como la controlada tienen responsabilidad tanto en los envíos de los datos como en la gestión (establecimiento y liberación) del nivel de enlace (fuera del alcance de las especificaciones del protocolo).

El empleo de doble direccionamiento (dirección de origen y dirección de destino) se debe a la funcionalidad que proporciona DNP3 basado en funcionamiento por excepción. De tal modo las comunicaciones no son iniciadas únicamente por la estación controladora, enviando preguntas a las estaciones controladas, sino que además estas últimas pueden iniciar una conversación dependiendo de la alteración de determinada información configurada en ella para ser reportada en estas condiciones. A este tipo de mensajes, en los cuales la estación controlada transmite los eventos de determinados objetos configurados en ella, se les denomina "respuestas no solicitadas".

El nivel de enlace proporciona una serie de servicios para la gestión de la comunicación entre las estaciones, tales como la petición o envío con o sin confirmación, las confirmaciones de tramas recibidas (ACK), las confirmaciones negativas (NACK), el reset de enlace (Reset Link) o el chequeo del estado del enlace (Link Status) [12].

3.3 Conceptos generales utilizados para la descripción del protocolo DNP3.

Respecto a la Capa de Usuario, la literatura de DNP3 utiliza el término Punto (point) o tag para asociar una entrada/salida analógica o binaria, o bien a un contador con el valor específico de su medición como se muestra en la Figura 19.
Los puntos se clasifican de acuerdo a sus características de funcionalidad, relación con el hardware o espacio lógico asociado. En DNP3 cada tipo de Punto se visualiza como un arreglo de Puntos independientes e indexados, en donde cada Punto es único e identificable. Además, se deben considerar las siguientes características:

1. Cada Punto es identificable dentro del arreglo.
2. Un Punto es usualmente un valor estático.
3. Los puntos pueden generar eventos.

Para identificar un Punto se utilizan un índice (index) y un grupo (group), y para definir el tipo de datos se utiliza una variación (variation); una Estación Remota puede transportar cinco tipos de puntos como muestra la Figura 19, y en algunos casos en particular soporta la transmisión de archivos y otros tipos de datos [14].

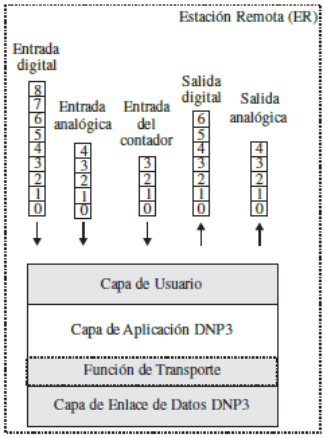

Figura 19. 5 tipos de puntos. Entradas analógicas, digitales y de contador y salidas digitales y analógicas [14].

DNP3 utiliza índices para identificar a los puntos que son del mismo tipo, los números de los índices corresponden a los números de los elementos del arreglo. Los grupos clasifican el tipo o tipos de datos que contiene el mensaje; cada grupo indica las mediciones del mismo tipo y el método de generación de datos. La Figura 20 lista la clasificación de grupos DNP3 [14].

3.3.1 Grupo.

El grupo especifica el tipo de dato o valores solicitados por la estación maestra o en una respuesta de la estación remota. Algunos ejemplos son: Entradas analógicas, valores actuales, entradas binarias, valores de eventos, Contador congelado, valor actual o valores de los contadores.

Grupo	Descripción
30	Valor actual de la medición
31	Valor fijo de la medición
32	Cambio del valor actual del evento
33	Cambio en el valor fijo del evento

Figura 20. Grupos de datos (grupos de objetos) [14].

Ocasionalmente, se utiliza el número de grupo para especificar el tipo de datos que se reporta (valor del temporizador, archivos de control, características de terminales virtuales, etcétera). DNP3 soporta diferentes tipos de datos, a esto se le conoce como variación, y cada grupo puede elegir entre las variaciones de la Figura 21.

3.3.2 Variación.

La variación especifica el formato de datos de los objetos DNP3. Por ejemplo, las

variaciones existentes para informar sobre el valor actual de las entradas analógicas como enteros de 16 bits, enteros de 32 bits, las cantidades de decimales de un número flotante, los enteros pueden incluir un byte adicional para banderas. Cada formato tiene distintas alternativas de variación única.

Variación	Descripción
1	Entero de 32 bits con bandera
2	Entero de 16 bits con bandera
3	Entero de 32 bits
4	Entero de 16 bits
5	Flotante de 32 bits con bandera
6	Flotante de 64 bits con bandera

Figura 21. Variaciones. (Tipos de registros) [14].

3.4 Instrucciones de protocolo DNP soportadas por medidores SHARK.

Debido a que no se usa todo el protocolo DNP3 en los medidores SHARK, al formato de protocolo DNP en estos medidores se les conoce como DNP lite debido a que el protocolo DNP usado es un subconjunto reducido del protocolo DNP 3.0. Aun así tiene las suficientes funcionalidades para ser interrogado por un maestro [9].

El DNP Lite soporta solo objeto clase 0. Ninguna generación evento es soportado. En otras palabras, no soporta objetos de clase 1,2 y 3, y este en protocolo DNP Lite siempre actuará como un dispositivo secundario (esclavo).

En cuanto al medio físico, DNP Lite utiliza la comunicación serial. Puede ser asignado al Port 2 (Puerto compatible RS-485) o cualquier otra comunicación opcional capaz. La velocidad y formato de datos es transparente para el DNP Lite: se pueden establecer en cualquier valor admitido. Los Puertos IrDA y TCP/IP, no pueden utilizar DNP Lite.

En cuanto a la capa de Enlace de Datos, a los medidores SHARK se les puede asignar un valor de 1 a 65534 como la dirección del dispositivo para el DNP Lite aunque la pantalla del medidor, permite configurarlo manualmente entre 0 y 999.

Los medidores SHARK, DNP soportan la función de Lectura, la función Escritura, la función Operar Directamente y la función Operación Directa sin Confirmar.

La función de Lectura (código 01) proporciona un medio para la lectura de los datos críticos de medición desde el medidor que muestra una lectura de datos de clase 0 solicitada desde el maestro.

La función Escribir (código 02) constituye un medio más para eliminar el bit de reinicio del dispositivo solamente en el indicador interno de registro. Esta se asigna a objetos 80, punto 0 con variación 1. Esta función sirve para desde el maestro darle nada más una indicación de reinicio al medidor.

La función Operación Directa (código 05) se destina para restablecer los contadores de energía y los contadores de demanda (los registros de energía mínimo y máximo). Estas

Acciones se asignan a objetos 12, punto 0 y punto 2, que son vistos como un relé de control. El relé debe ser operado (On) en 0 ms y liberarlo (Off) en sólo 1 mseg.
La Operación Directa sin Confirmar (o no reconocidos) función (código 06) se destina para pedir el puerto de comunicación para cambiar al protocolo Modbus RTU desde el DNP. Este cambio es visto como un relé de control asignado en objeto 12, punto 1 en el medidor.

Después de enviar esta petición, el puerto de comunicación actual aceptará marcos Modbus RTU solamente. Para que este puerto se regrese al protocolo DNP, la unidad debe ser reenergizada.

En el caso de una función no soportada, o cualquier otro error reconocible, una respuesta de error será generada desde medidor SHARK desde la estación de primaria (el solicitante).

La Figura 22 muestra la trama de datos general de como vienen los datos al hacer un análisis de integridad al medidor, y la obtención de los valores analógicos del objeto 30 y variación 4, viniendo en la porción de “reply”, todos los puntos correspondientes a las medidas analógicas del medidor.

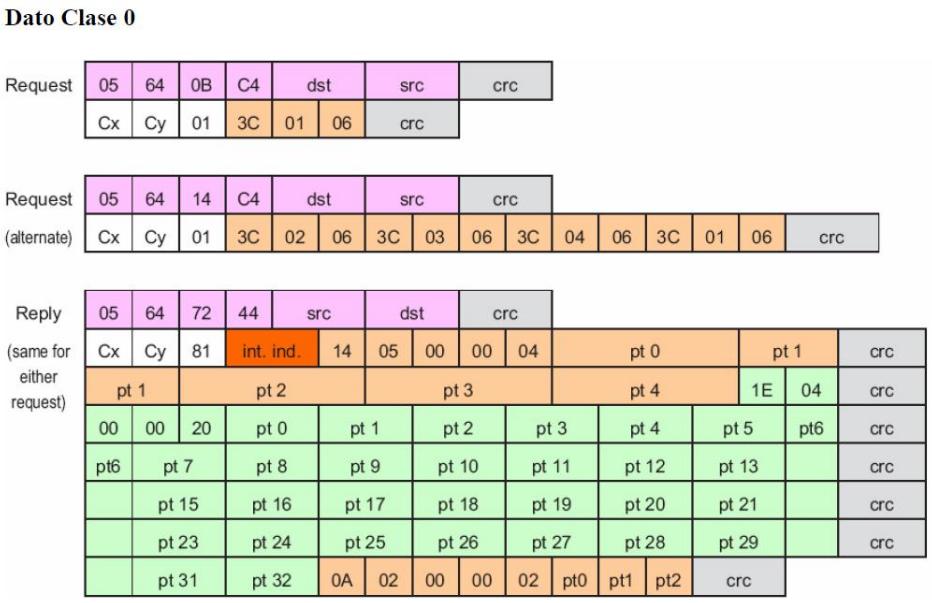

Figura 22. Ejemplo de solicitud-respuesta de medidores SHARK descrita en el apéndice C [9].

Este formato de respuesta, se ve en el apartado de simulación, en el que el simulador de OpenDNP3, devuelve los datos medidos en hexadecimal y mapeados desde 0 hasta 32768 según el mapa DNP. El mapa DNP se muestra en el anexo D de este trabajo.

3.5 Interrogación con simulador de Opendnp3.

Opendnp3 posee un simulador el cual puede descargarse desde [6], Con el cual puede establecerse unidades maestro y estaciones remotas. Este simulador está diseñado para ser instalado en sistema operativo Windows. Una característica importante de este

simulador, es que puede establecerse un maestro, y establecerse un canal de comunicación que puede ser TCP/IP o Serie (por medio de un puerto RS485). En nuestro caso es necesario establecer el canal de comunicación Serie y tener instalado en la computadora el driver del conector USB-RS485. La comunicación únicamente se realiza mediante el protocolo DNP3.

3.5.1 Configuración del simulador.

Primero debe procederse a la instalación del software y al estar instalado se abre la interfaz la cual es como se muestra en la Figura 23.

Figura 23. Interfaz principal de software de OpenDNP3.

Principalmente se cierra la ventana About, y luego se debe establecer el canal de comunicación. Para esto debemos principalmente insertar el dispositivo USB-RS485 en el computadora, y debe notificarse al usuario que el dispositivo ha sido instalado correctamente y en que puerto “COM” ha sido insertado. Esa confirmación puede ser como se muestra en la Figura 24.

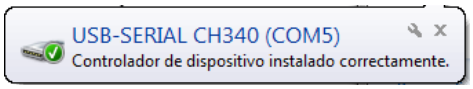

Figura 24. Reconocimiento correcto de dispositivo USB-RS485 en windows.

Estando instalado el dispositivo debemos tener en cuenta que en este caso el conector está ubicado en el puerto COM5. Ya ubicado el usuario en la interfaz del simulador, debe elegir Add y DNP3 Channel como se observa en la Figura 25.

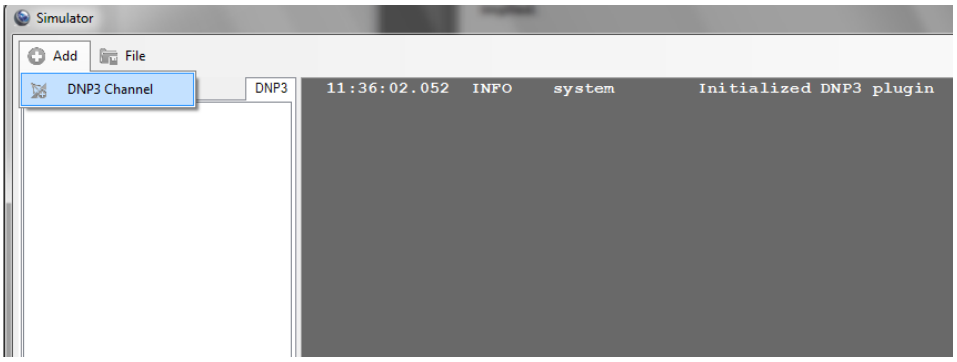

Figura 25. Añadiendo canal de comunicación.

Aparecerá un recuadro en el que se nos dará la opción de añadir un Servidor TCP, un Cliente TCP y un canal Serial. Nosotros elegiremos el canal Serial. Desplegaremos la pestaña Port, y elegiremos el puerto correspondiente (COM5) tal como se muestra en la Figura 26.

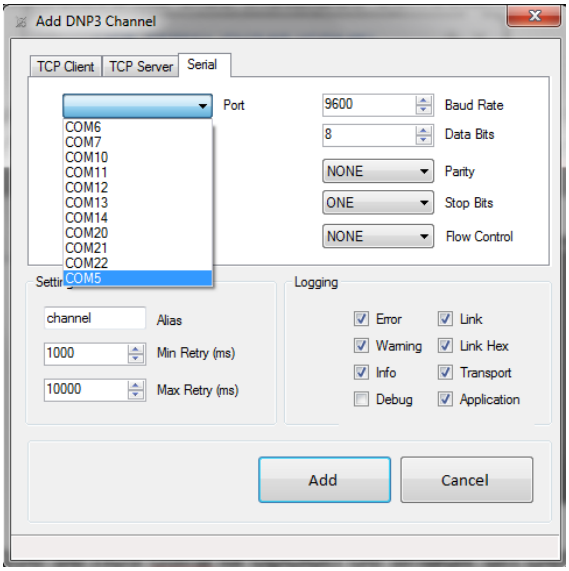

Figura 26. Sintonización del canal configurado en el medidor.

Es importante aclarar que el simulador, por defecto tiene un Baud Rate de 9600 y por lo tanto, el medidor, se debe también tener configurado a ese número. En caso de variarse ese valor en el simulador, debe hacerse también el cambio en el medidor como se detalló en la sección 2.2. El nombre (Alias), puede cambiarse a gusto del usuario.

Ahora que el canal de comunicación esta ya creado, se debe establecer la estación maestra. Entonces: Hacemos clic derecho sobre cannel y elegimos Add master como se muestra en la Figura 27.

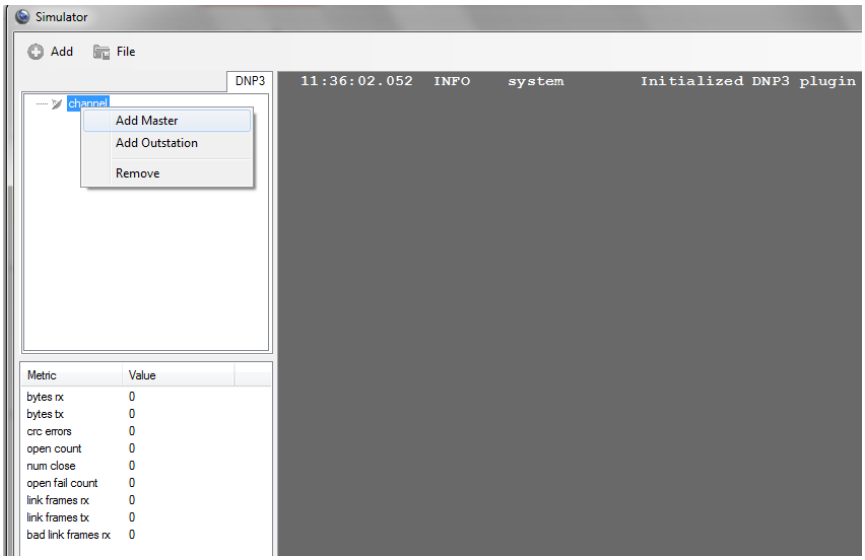

Figura 27. Simulación de un maestro que interrogara a la estación remota (medidor SHARK 200).

En la pestaña Link, debemos colocar la dirección serie del maestro (generalmente “1”), y la dirección de la estación remota, que es la que se debe configurar manualmente en el medidor como se muestra en la sección 2.2. Una vez configurado esto, se puede proceder la pestaña Master en el simulador como se muestra en la Figura 28, en la que se puede cambiar el nombre del maestro por uno de la preferencia del usuario y en la cual, se observa que se realizaría un escaneo integral (en el que se escanean todas las clases: 0, 1, 2 y 3). Y también se tiene habilitado el campo de las respuestas no solicitadas de clase 1, 2 y 3. Cabe destacar que el medidor SHARK, puede ser interrogado nada más por clase 0, como se detalla en la sección 3.4.

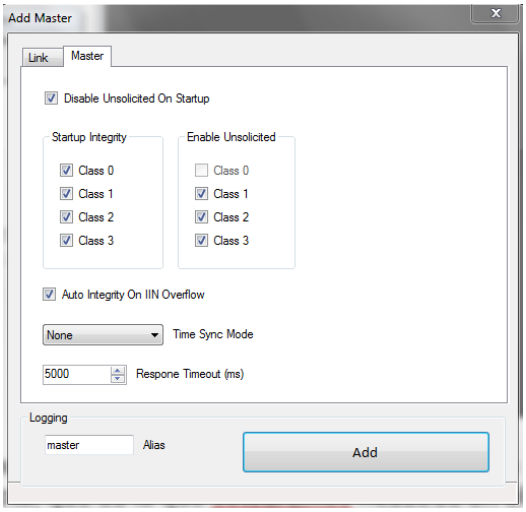

Figura 28. Estableciendo el Alias del maestro y operaciones que se realizaran.

Al finalizar la configuración, se presiona el botón Add e iniciará inmediatamente la comunicación entre el maestro que se está simulando y la estación remota que es el medidor. En la Figura 29 se muestran los datos devueltos por el simulador.

```
14:26:34.045 INFO   system      Initialized DNP3 plugin
14:26:51.922 INFO   master      Begining task: Disable Unsolicited
14:26:51.922 -AL->  master      FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 0 FUNC: DISABLE_UNSOLICITED
                    060,002 - Class Data - Class 1 data - all objects
                    060,003 - Class Data - Class 2 data - all objects
                    060,004 - Class Data - Class 3 data - all objects

14:26:51.922 -TL->  master      FIR: 1 FIN: 1 SEQ: 0 LEN: 11
14:26:51.922 -LL->  master      Function: PRI_UNCONFIRMED_USER_DATA Dest: 2 Source: 1 Length: 12
                    05 64 11 C4 02 00 01 00 29 E0
                    C0 C0 15 3C 02 06 3C 03 06 3C 04 06 1A 55

14:26:51.985 <-LL-  channel     Function: PRI_UNCONFIRMED_USER_DATA Dest: 1 Source: 2 Length: 10
                    05 64 0A 44 01 00 02 00 FA 4A
                    C0 C0 81 00 01 C2 DE

14:26:51.985 <-TL-  master      FIR: 1 FIN: 1 SEQ: 0 LEN: 4
14:26:51.985 <-AL-  master      FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 0 FUNC: RESPONSE IIN: [0x00, 0x01]
14:26:51.985 WARN   master      Task was explicitly rejected via response with error IIN bit(s): Disable
Unsolicited
14:26:52.000 INFO   master      Begining task: Application Poll
14:26:52.000 -AL->  master      FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 1 FUNC: READ
                    060,002 - Class Data - Class 1 data - all objects
                    060,003 - Class Data - Class 2 data - all objects
                    060,004 - Class Data - Class 3 data - all objects
                    060,001 - Class Data - Class 0 data - all objects
14:26:52.000 -TL->  master      FIR: 1 FIN: 1 SEQ: 1 LEN: 14
14:26:52.000 -LL->  master      Function: PRI_UNCONFIRMED_USER_DATA Dest: 2 Source: 1 Length: 15
                    05 64 14 C4 02 00 01 00 A0 18
                    C1 C1 01 3C 02 06 3C 03 06 3C 04 06 3C 01 06 62 01
14:26:52.172 <-LL-  channel     Function: PRI_UNCONFIRMED_USER_DATA Dest: 1 Source: 2 Length: 114
                    05 64 72 44 01 00 02 00 18 F7
                    C1 C1 81 00 00 14 05 00 00 04 E8 60 16 00 FA F6 B1 7D
                    00 00 11 43 11 01 49 FF F1 05 2A E7 1D 02 1E 04 9B AC
                    00 00 20 00 00 31 64 00 00 9D 08 15 32 4E 04 2F 0C CE
                    2E 89 00 00 00 00 00 F9 FF DD FF 24 00 40 FF 6F C7 E0
                    17 5E 55 28 50 37 E7 45 A6 34 66 F2 03 00 00 00 37 81
                    00 50 FB 84 05 14 FF 90 01 01 00 05 00 D0 00 01 79 C9
                    00 D0 00 89 00 0A 02 00 00 02 01 01 01 32 17
14:26:52.172 <-TL-  master      FIR: 1 FIN: 1 SEQ: 1 LEN: 108
14:26:52.172 <-AL-  master      FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 1 FUNC: RESPONSE IIN: [0x00, 0x00]
                    020,005 Counter - 32-bit Without Flag, 8-bit start stop [0, 4]
                    030,004 Analog Input - 16-bit Without Flag, 8-bit start stop [0, 32]
                    010,002 Binary Output - Output Status With Flags, 8-bit start stop [0, 2]

14:26:52.250 INFO   master      Begining task: Enable Unsolicited
14:26:52.250 -AL->  master      FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 2 FUNC: ENABLE_UNSOLICITED
```

```
                    060,002 - Class Data - Class 1 data - all objects
                    060,003 - Class Data - Class 2 data - all objects
                    060,004 - Class Data - Class 3 data - all objects

14:26:52.250 -TL->  master      FIR: 1 FIN: 1 SEQ: 2 LEN: 11
14:26:52.250 -LL->  master      Function: PRI_UNCONFIRMED_USER_DATA Dest: 2 Source: 1 Length: 12
                    05 64 11 C4 02 00 01 00 29 E0
                    C2 C2 14 3C 02 06 3C 03 06 3C 04 06 D3 2F

14:26:52.312 <-LL-  channel     Function: PRI_UNCONFIRMED_USER_DATA Dest: 1 Source: 2 Length: 10
                    05 64 0A 44 01 00 02 00 FA 4A
                    C2 C2 81 00 01 67 50

14:26:52.312 <-TL-  master      FIR: 1 FIN: 1 SEQ: 2 LEN: 4
14:26:52.312 <-AL-  master      FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 2 FUNC: RESPONSE IIN: [0x00, 0x01]

14:26:52.312 WARN   master      Task was explicitly rejected via response with error IIN bit(s): Enable
Unsolicited
```

Figura 29. Trama de datos obtenida, debido a la interrogación del medidor con el simulador Opendnp3.

En la Figura 29 en la porción marcada en rojo se muestra la confirmación de la respuesta de datos del grupo de objetos 30 y variación 4 donde aparecen todos los puntos mapeados y en su valor hexadecimal respectivamente. La porción de referencia está en la sección 3.4 específicamente en la Figura 22.

3.6 Implementación de código en lenguaje c++ para interrogación de medidores por DNP3.

El procedimiento que a continuación debe seguirse, es el de implementar un código con la sintaxis de la biblioteca OpenDNP3 en este caso en el lenguaje c++, que se encargue de interrogar al medidor de energía. Siendo que en este caso no se busca modificar el código CMakeLists.txt, lo que se hará es, colocarle el nombre “main.cpp” al código realizado, y sustituirlo por el código main.cpp que se encuentra en la carpeta “master” de la biblioteca, específicamente en la dirección /dnp3/cpp/examples/master/.

Posteriormente al tener sustituido el código, ingresamos a la terminal de linux. En ese momento es necesario volverse ¨root¨ nos volvemos “root” con el siguiente comando.

```
root@jose-Latitude-D510:/home/jose# sudo su
```

En ese momento debe escribirse la clave de superusuario. Siendo root, el usuario debe ubicarse en el directorio dnp3, y proceder a la compilación de la biblioteca. Para ello se escribe la línea.

```
root@jose-Latitude-D510:/home/jose# make
```

Y automáticamente iniciara la compilación. En caso de existir algún error, mediante la compilación, este se notificara en la misma terminal. En ese momento se debe proceder a la corrección del código y al estar corregido, se debe guardar los cambios y volver a compilar.

A continuación en la Figura 30, se muestra el código que se implementó, con las respectivas explicaciones de las instrucciones más importantes.

```
1. #include <asiodnp3/DNP3Manager.h>
2. #include <asiodnp3/PrintingSOEHandler.h>
3.
4. #include <asiodnp3/ConsoleLogger.h>
5. #include <asiodnp3/DefaultMasterApplication.h>
6. #include <asiodnp3/PrintingCommandCallback.h>
7. #include <asiodnp3/PrintingChannelListener.h>
8. #include <asiopal/UTCTimeSource.h>
9. #include <opendnp3/LogLevels.h>
10.     #include <chrono>
11.     #include <thread>
12.
13.     using namespace std::this_thread; // sleep_for, sleep_until
14.     using namespace std::chrono;
15.     using namespace std;
16.     using namespace openpal;
17.     using namespace asiopal;
18.     using namespace asiodnp3;
19.     using namespace opendnp3;
20.     int main(){
21.             const uint32_t FILTERS = levels::NORMAL | levels::ALL
   _APP_COMMS;
22.             DNP3Manager manager(1, ConsoleLogger::Create());
23.             SerialSettings settings;
24.             settings.deviceName = "/dev/ttyUSB0";
25.             auto channel = manager.AddSerial("chepin", FILTERS,
   ChannelRetry::Default(), settings,
   PrintingChannelListener::Create());
26.
27.             MasterStackConfig stackConfig;
28.             stackConfig.master.responseTimeout = TimeDuration::Se
   conds(2);
29.             stackConfig.master.disableUnsolOnStartup = true;
30.             stackConfig.link.LocalAddr = 1;
31.             stackConfig.link.RemoteAddr = 2;
32.
```

```
33.             auto master = channel-
   >AddMaster("master",PrintingSOEHandler::Create(),
   asiodnp3::DefaultMasterApplication::Create(),stackConfig);
34.             master->Enable();
35.             master->ScanRange(GroupVariationID(30, 4), 0, 32);
36.
37.             sleep_for(seconds(1));
38.             return 0;
39.     }
```

Figura 30. Código para interrogación preliminar con DNP3 RTU.

El código realiza la función de interrogar el grupo de objetos 30 y la variación 4. Lo cual indica, que hace una lectura de todos los datos analógicos y muestra los que el usuario solicita con la función “Scan Range”. En este caso se ha solicitado los 33 puntos disponibles de las entradas analógicas del mapa DNP de los medidores SHARK. (Vease el mapa DNP en el anexo 4 de este documento).

Las líneas 1 - 11 especifican las librerías incluidas en el código de la figura 30 para que este incluya los códigos correspondientes de esas líneas y poder usar funciones específicas descritas también en esos códigos. Las líneas 13 - 19, contienen las palabras reservadas “Using namespace…” que deben especificarse para poder usar determinadas funciones pertenecientes a las librerías.

Las líneas 21 - 25 especifican el tipo de canal que se creara. En nuestro caso nos interesa crear un canal Serie. En la línea 24 se especifica el nombre con el cual reconoce la computadora con sistema operativo Linux al convertidor USB-RS485. Pero en caso de usar otro dispositivo, solo debe verificarse en la carpeta /dev del equipo cual es el nombre que la computadora le da al dispositivo para poder incluirlo en el programa y que este sea reconocido.

Las líneas 27 - 31 se asignan datos como: Tiempos de respuesta, activación/desactivación del modo “respuesta no solicitada” entre otros. Los valores más trascendentales se encuentran entre las líneas 30 y 31. La línea 30 se le coloca la dirección de enlace específica del maestro. Generalmente se le coloca el número de dirección “1”, pero puede colocársele cualquier dirección que no sea igual a la del medidor. En la línea 31 se observa la dirección del medidor. Esa dirección en el caso de los medidores se configura como se muestra en la sección 2.2.

En caso de no estar sintonizados los valores, de las direcciones en la capa de enlace de datos, al momento de correr el programa, se mostrara un error como el que se muestra en la Figura 31.

```
root@jose-Latitude-D510:/home/jose/Escritorio/prueba/dnp3# ./master-demo
ms(1537308855254) INFO    manager - Starting thread (0)
channel state change: OPENING
ms(1537308855255) WARN    Serial - Error Connecting: No such file or directory
ms(1537308856255) WARN    Serial - Error Connecting: No such file or directory
channel state change: SHUTDOWN
ms(1537308856255) INFO    manager - Exiting thread (0)
```

Figura 31. Error de comunicación debido a datos no sincronizados entre maestro y estación remota.

Este error también ocurrirá, si no se especifica correctamente, o en su defecto, no se especifica el nombre del convertidor USB-RS485 en el programa.

La línea 33 define las características del maestro. Entre ellas cuál será su nombre y demás parámetros que están especificados en otros códigos de la librería que se han llamado y son parte de “DNP3Manager”.

Las líneas 34 y 35 habilitan el maestro (línea 34) para se pueda hacer un escaneo en el rango de interés (línea 35). En este código, nada más se interroga la trama de datos en el rango de puntos que se requiera. La línea 37 permite que el equipo espere por un segundo mediante se realiza la interrogación y la devolución de los datos. En caso de no permitirse este margen, el resultado no se devuelve al maestro y no se muestra en la línea de comandos, porque el canal se cierra antes que la respuesta llegue al maestro. Las últimas líneas, muestran el retorno de la función main y el fin del programa.

En el caso de querer interrogar por DNP3 TCP/IP, se debe realizar un cambio entre las líneas 23 y 25 en la que se le especifique al programa las IP del maestro y la IP de la estación remota como se muestra a continuación en la Figura 32.

```
23.     auto channel = manager.AddTCPClient("tcpclient",FILTERS,Chann
   elRetry::Default(),    "127.0.0.1",        "0.0.0.0",        20000,
   PrintingChannelListener::Create());
```

Figura 32. Línea por la cual deben ser sustituidas las características del puerto seria, para comunicación TCP/IP.

El bloque que contiene la IP “127.0.0.1” corresponde al maestro. Debe colocarse específicamente ya sea la IP, o el alias asignado al maestro. Y en el bloque donde está la IP “0.0.0.0”, debe colocarse la IP del medidor. Puntos importantes a destacar, son, que el medidor debe encontrarse en el rango del maestro, y que el medidor debe estar configurado para DNP3 TCP respectivamente. Los medidores SHARK no poseen la capacidad de comunicarse mediante DNP3 por TCP.

3.7 Interrogación de medidores y realización de ejemplos.

3.7.1 Ejemplo 1. Interrogación correspondiente con código de sección 3.6.

Si se encuentra todo en orden y se han realizado los procedimientos de instalación y configuración del anexo A y B correctamente procedemos a compilar la biblioteca con la siguiente línea.

```
root@jose-Latitude-D510:/home/jose# make
```

Al realizar la compilación del código, obtendremos 3 ejecutables en la misma carpeta "dnp3". El nombre específico del ejecutable de interés es "master-demo", debido a que así está especificado en el código CMakeList.txt del anexo C. Si se hace el cambio de ubicación, el código quedara en un lugar diferente y si se cambia el nombre, puede colocarsele el nombre que el usuario desee.

Es importante que antes de correr el código se le brinden los permisos necesarios al puerto serie para que este se habilite y de lugar a la comunicación. Eso se logra con la instrucción siguiente.

```
root@jose-Latitude-D510:/home/jose# chmod 666 /dev/tty/USB0
```

De la compilación, se generara el ejecutable "master-demo". Para correr el ejecutable, desde la línea de comandos debe escribirse la siguiente línea:

```
root@jose-Latitude-D510:/home/jose# ./master-demo
```

Si no hay ningún error y ocurre correctamente la consulta de los datos, en la terminal se mostrara el resultado de la interrogación de los 33 puntos del grupo de objetos 30 y variación 4, y se verán como se observa en la Figura 33.

```
ms(1537389244556) INFO    master - Begining task: Enable Unsolicited
ms(1537389244556) --AL->  master - C2 14 3C 02 06 3C 03 06 3C 04 06
ms(1537389244556) --AL->  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 2 FUNC: ENABLE_UNSOLICITED
ms(1537389244556) --AL->  master - 060,002 - Class Data - Class 1 - all objects
ms(1537389244556) --AL->  master - 060,003 - Class Data - Class 2 - all objects
ms(1537389244556) --AL->  master - 060,004 - Class Data - Class 3 - all objects
ms(1537389244602) <-AL--  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 2 FUNC: RESPONSE IIN: [0x00, 0x01]
ms(1537389244603) WARN    master - Task was explicitly rejected via response with error IIN bit(s): Enable
Unsolicited
ms(1537389244603) INFO    master - Begining task: Application Poll
```

```
ms(1537389244603) --AL->  master - C3 01 1E 04 01 00 00 20 00
ms(1537389244603) --AL->  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 3 FUNC: READ
ms(1537389244603) --AL->  master - 030,004 Analog Input - 16-bit Without Flag, 16-bit start stop [0, 32]
ms(1537389244727) <-AL--  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 3 FUNC: RESPONSE IIN: [0x00, 0x00]
ms(1537389244728) <-AL--  master - 030,004 Analog Input - 16-bit Without Flag, 8-bit start stop [0, 32]

[0] : 0
[1] : 25620
[2] : 0
[3] : 0
[4] : 12810
[5] : 0
[6] : 12810
[7] : 51
[8] : 0
[9] : 0
[10] : 0
[11] : -14
[12] : 14
[13] : -30
[14] : 6001
[15] : 21854
[16] : 20520
[17] : -6345
[18] : -22971
[19] : 26164
[20] : 917
[21] : 0
[22] : 0
[23] : -1258
[24] : 269
[25] : 989
[26] : 400
[27] : 1
[28] : 5
[29] : 208
[30] : 1
[31] : 208
[32] : 51

channel state change: CLOSED
channel state change: SHUTDOWN
```

```
ms(1537389245309) INFO    manager - Exiting thread (0)
root@jose-Latitude-D510:/home/jose/Escritorio/prueba/dnp3#
```

Figura 33. Obtención de diferentes lecturas de datos mediante interrogación preliminar DNP3.

El valor correspondiente al punto 1, es el voltaje en la fase A según el mapa DNP mostrado en el anexo D. La razón por la que aparece un valor no coherente es porque el valor esta escalado.

El medidor hace muestreo de 16 bits, por lo cual. Por ejemplo el valor de voltaje de una sola fase, según el mapa DNP del medidor SHARK 200S no estará mostrado entre 0 y 150 volts, sino estará mapeado en la escala de 0 a 2^{16} $(65{,}536)$. En este caso específico, el valor de 65536 se parte en 2 mostrándose la manera negativa desde -32768 hasta 0, y la parte positiva, desde 0 hasta 32768. Por ejemplo. Al observar la Figura 34, La porción encerrada en amarillo, son los datos correspondientes al voltaje en la fase A según el mapa DNP.

Object	Point	Var	Description	Format	Range	Multiplier	Units	Comments
30	0	4	Meter Health	sint16	0 or 1	N/A	None	0 = OK
30	1	4	Volts A-N	sint16	0 to 32767	(150 / 32768)	V	Values above 150V secondary read 32767.
30	2	4	Volts B-N	sint16	0 to 32767	(150 / 32768)	V	

Figura 34. Porción de mapa DNP en la que se observan los datos correspondientes a Voltaje en la fase A [9].

En la porción del mapa DNP, se muestra que se debe aplicar el multiplicador 150/32768 al valor muestreado por el medidor y mostrado en la terminal en el punto 1 específicamente. Entonces Hacemos el cálculo y el valor real de voltaje es:

$$Voltaje\ A = 25826\frac{150}{32768} = \ 118.22\ V$$

Entonces, en el siguiente ejercicio la propuesta es, realizar lo necesario para poder aplicar dicho factor de conversión, para poder mostrar en la terminal de Linux, el valor analógico correspondiente.

3.7.2 Ejemplo 2. Impresión de valores analógicos reales en la terminal.

Para poder mostrar los valores reales de cada lectura hay que aplicar el factor

correspondiente a cada punto como el que se mostró en la sección 3.7.1 para el voltaje. Para poder realizar esto, tenemos un problema. Ocurre que la impresión del valor que se muestra en la terminal, no se puede manipular en el programa principal (main.cpp) de la Figura 30. En ese programa, la función dada a “master” en la línea 47 del programa, implica, que el maestro, realizara el escaneo en el medidor del grupo de objetos 30 y la variación 4, recibe a la vez los datos y los imprime en pantalla acompañado del número correlativo. Sin embargo la biblioteca está diseñada, para que esa función envíe los datos a otros códigos de la biblioteca, siendo el más importante “ISOEhandler.h”. Este a su vez procesa los datos hacia una dependencia llamada “printingSOEhandler.h” el cual es el encargado de la impresión general de los datos recibidos de toda la biblioteca, sean estos binarios, analógicos, contadores, etc.

Este programa se encuentra en la ubicación /dnp3/cpp/libs/include/asiodnp3/ de la biblioteca. También en el directorio /dnp3/cpp/libs/src/asiodnp3/ existe un código llamado printingSOEhandler.cpp que es el que se encarga de enviar los datos a printingSOEhandler.h para su posterior impresión.

Uno de los problemas con printingSOEhandler.cpp es que ese código envía también información de grupos de objetos que no necesitamos. Por lo cual hay que realizar cambios también para que se adecue a nuestro interés. (La interrogación de las entradas analógicas del medidor).

Ocurre que si se desea editar esos archivos, la librería seria configurada. Y en caso de quererse reinstalar la librería, se descargaran nuevamente los archivos ya mencionados, de manera intacta, habiéndose perdido los cambios que nos permitirían en este caso, observar los datos de forma analógica. En ese caso, habría que volver a hacer los cambios en cada código correspondiente lo cual carece de flexibilidad y se vuelve tedioso. También se debe valorar que al hacerse ese proceso, si llegara a ocurrir errores de compilación por no copiarse correctamente una línea, un signo entre otras cosas habrá problemas que pueden ser serios. Por lo cual se procede a hacer lo siguiente.

3.7.2.1 sustituciones de códigos J_PrintingSOEhandler.

Se procedió a crear un nuevo codigo en sustitución de printingSOEhandler.h y printingSOEhandler.cpp respectivamente. A los cuales se les llamo J_printingSOEhandler.h el cual debe ser guardado en la dirección /dnp3/cpp/libs/include/asiodnp3/ y J_printingSOEhandler.cpp el cual debe ser guardado en /dnp3/cpp/libs/src/asiodnp3/.

A continuación en la Figura 35, mostramos como finalmente debe estar reescrito el código J_printingSOEhandler.cpp.

```
1. #include "asiodnp3/J_PrintingSOEHandler.h"
2. using namespace std;
3. using namespace opendnp3;
4. namespace asiodnp3
5. {
6. void J_PrintingSOEHandler::Process(const HeaderInfo& info, const IC
   ollection<Indexed<Binary>>& values){}
7. void J_PrintingSOEHandler::Process(const HeaderInfo& info, const IC
   ollection<Indexed<DoubleBitBinary>>& values){}
8. void J_PrintingSOEHandler::Process(const HeaderInfo& info, const IC
   ollection<Indexed<Analog>>& values)
9. {
10.             return PrintAll(info, values);
11.     }
12.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<Counter>>& values){}
13.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<FrozenCounter>>& values){}
14.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<BinaryOutputStatus>>& values){}
15.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<AnalogOutputStatus>>& values){}
16.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<OctetString>>& values){}
17.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<TimeAndInterval>>& values){}
18.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<BinaryCommandEvent>>& values){}
19.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<AnalogCommandEvent>>& values){}
20.     void J_PrintingSOEHandler::Process(const HeaderInfo& info, co
   nst ICollection<Indexed<SecurityStat>>& values){}
21.
22.     void J_PrintingSOEHandler::Process(const opendnp3::HeaderInfo
   & info, const opendnp3::ICollection<opendnp3::DNPTime>& values){}
23.     }
```

Figura 35. Código J_printingSOEhandler.cpp que debe ubicarse en /dnp3/cpp/libs/src/asiodnp3/.

Lo que cabe resaltar en el código de la Figura 35, es que los únicos valores que retornan hacia printingSOEhandler.h para ser impresos, son los datos correspondientes a entradas analógicas del medidor recolectados éntrelas líneas 8 y 11. Siendo así, los valores correspondientes a los demás grupos de objetos, ya no tendrán posibilidad de mostrarse en esta aplicación específica.

A continuación en la Figura 36 se muestra el código correspondiente a J_printingSOEhandler.h.

```
1. #ifndef ASIODNP3_PRINTINGSOEHANDLER_H
2. #define ASIODNP3_PRINTINGSOEHANDLER_H
3. #include <opendnp3/master/ISOEHandler.h>
```

```
4. #include <iostream>
5. #include <sstream>
6. #include <memory>
7. #include <string>
8. #include <fstream>
9. #include <usr/include/mysql++/mysql++.h>
10.     namespace asiodnp3
11.     {
12.     class J_PrintingSOEHandler final : public opendnp3::ISOEHandl
   er
13.     {
14.     public:
15.             J_PrintingSOEHandler()
16.             {}
17.             static std::shared_ptr<ISOEHandler> Create()
18.             {
19.                     return std::make_shared<J_PrintingSOEHandler>
   ();
20.             }
21.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::Binary>>&
    values) override;
22.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::DoubleBit
   Binary>>& values) override;
23.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::Analog>>&
    values) override;
24.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::Counter>>
   & values) override;
25.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::FrozenCou
   nter>>& values) override;
26.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::BinaryOut
   putStatus>>& values) override;
27.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::AnalogOut
   putStatus>>& values) override;
28.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::OctetStri
   ng>>& values) override;
29.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::TimeAndIn
   terval>>& values) override;
30.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::BinaryCom
   mandEvent>>& values) override;
31.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::AnalogCom
   mandEvent>>& values) override;
```

```
32.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<opendnp3::SecurityS
   tat>>& values) override;
33.             virtual void Process(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::DNPTime>& values) override;
34.
35.     protected:
36.
37.             void Start() final {}
38.             void End() final {}
39.
40.     private:
41.
42.             template <class T>
43.             static void PrintAll(const opendnp3::HeaderInfo& info
   , const opendnp3::ICollection<opendnp3::Indexed<T>>& values)
44.             {
45.                     auto print = [&](const opendnp3::Indexed<T>&
   pair)
46.                     {
47.                             Print<T>(info, pair.value,
   pair.index);
48.                     };
49.                     values.ForeachItem(print);
50.             }
51.
52.             template <class T>
53.             static void Print(const opendnp3::HeaderInfo& info, c
   onst T& value, uint16_t index)
54.             {
55.                     std::ofstream archivo;
56.                     std::string jose = ValueToString(value);
57.                     float a;
58.                     std::istringstream(jose) >> a;
59.                     switch(index)
60.                     {
61.                     case 0:
62.                             {
63.                              std::cout <<"\n"<<"Salud del
   medidor:\t\t"<< "[" << index << "] : " << a << std::endl;
64.
65.         ``                  }
66.                     case 1:
67.                             {
68.                              a = (a*150)/32768;
69.                             std::cout <<"Voltaje
   A:\t\t\t"<< "[" << index << "] : " << a << std::endl;
70.                              break;
71.                             }
72.                     case 2:
73.                             {
74.                              a = (a*150)/32768;
```

```
75.                                 std::cout <<"Voltaje
   B:\t\t\t"<< "[" << index << "] : " << a << std::endl;
76.                                  break;
77.                                 }
78.                         case 3:
79.                                 {
80.                                  a = (a*150)/32768;
81.                                 std::cout <<"Voltaje
   C:\t\t\t"<< "[" << index << "] : " << a << std::endl;
82.                                  break;
83.                                 }
84.                         case 4:
85.                                 {
86.                                  a = (a*300)/32768;
87.                                 std::cout <<"Voltaje A-
   B:\t\t\t"<< "[" << index << "] : " << a << std::endl;
88.                                  break;
89.                                 }
90.                         case 5:
91.                                 {
92.                                  a = (a*300)/32768;
93.                                 std::cout <<"Voltaje B-
   C:\t\t\t"<< "[" << index << "] : " << a << std::endl;
94.                                  break;
95.                                 }
96.                         case 6:
97.                                 {
98.                                  a = (a*300)/32768;
99.                                 std::cout <<"Voltaje C-
   A:\t\t\t"<< "[" << index << "] : " << a << std::endl;
100.                                 break;
101.                                }
102.                        case 7:
103.                                {
104.                                 a = (a*10)/32768;
105.                                std::cout <<"Corriente
   A:\t\t\t"<< "[" << index << "] : " << a << std::endl;
106.                                 break;
107.                                }
108.                        case 8:
109.                                {
110.                                 a = (a*10)/32768;
111.                                std::cout <<"Corriente
   B:\t\t\t"<< "[" << index << "] : " << a << std::endl;
112.                                 break;
113.                                }
114.                        case 9:
115.                                {
116.                                 a = (a*10)/32768;
117.                                std::cout <<"Corriente
   C:\t\t\t"<< "[" << index << "] : " << a << std::endl;
118.                                 break;
119.                                }
```

```
120.                     case 10:
121.                             {
122.                              a = (a*4500)/32768;
123.                             std::cout <<"Watts, 3ph
   total:\t\t"<< "[" << index << "] : " << a << std::endl;
124.                              break;
125.                             }
126.                     case 11:
127.                             {
128.                              a = (a*4500)/32768;
129.                             std::cout <<"VARs, 3ph
   total:\t\t"<< "[" << index << "] : " << a << std::endl;
130.                              break;
131.                             }
132.                     case 12:
133.                             {
134.                              a = (a*4500)/32768;
135.                             std::cout <<"VAs, 3ph
   total:\t\t\t"<< "[" << index << "] : " << a << std::endl;
136.                              break;
137.                             }
138.                     case 13:
139.                             {
140.                              a = a*0.001;
141.                             std::cout <<"Factor de potencia 3ph
   total:\t"<< "[" << index << "] : " << a << std::endl;
142.                              break;
143.                             }
144.                     case 14:
145.                             {
146.                              a = a*0.01;
147.                             std::cout <<"Frecuencia:\t\t\t"<< "["
    << index << "] : " << a << std::endl;
148.                              break;
149.                             }
150.                     case 15:
151.                             {
152.                              a = (a*4500)/32768;
153.                             std::cout <<"Watts positivos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
154.                              break;
155.                             }
156.                     case 16:
157.                             {
158.                              a = (a*4500)/32768;
159.                             std::cout <<"VARs positivos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
160.                              break;
161.                             }
162.                     case 17:
163.                             {
164.                              a = (a*4500)/32768;
```

```
165.                                std::cout <<"Watts negativos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
166.                                 break;
167.                                }
168.                        case 18:
169.                                {
170.                                 a = (a*4500)/32768;
171.                                std::cout <<"VARs negativos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
172.                                 break;
173.                                }
174.                        case 19:
175.                                {
176.                                 a = (a*4500)/32768;
177.                                std::cout <<"VAs 3ph
   maximos:\t\t"<< "[" << index << "] : " << a << std::endl;
178.                                 break;
179.                                }
180.                        case 20:
181.                                {
182.                                 a = a*0.01;
183.                                std::cout <<"Angulo de fase.
   Corriente A:\t"<< "[" << index << "] : " << a << std::endl;
184.                                 break;
185.                                }
186.                        case 21:
187.                                {
188.                                 a = a*0.01;
189.                                std::cout <<"Angulo de fase.
   Corriente B:\t"<< "[" << index << "] : " << a << std::endl;
190.                                 break;
191.                                }
192.                        case 22:
193.                                {
194.                                 a = a*0.01;
195.                                std::cout <<"Angulo de fase.
   Corriente C:\t"<< "[" << index << "] : " << a << std::endl;
196.                                 break;
197.                                }
198.                        case 23:
199.                                {
200.                                 a = a*0.01;
201.                                std::cout <<"Angulo. Voltaje A-
   B:\t\t"<< "[" << index << "] : " << a << std::endl;
202.                                 break;
203.                                }
204.                        case 24:
205.                                {
206.                                 a = a*0.01;
207.                                std::cout <<"Angulo. Voltaje B-
   C:\t\t"<< "[" << index << "] : " << a << std::endl;
208.                                 break;
209.                                }
```

```
210.                     case 25:
211.                             {
212.                              a = a*0.01;
213.                             std::cout <<"Angulo. Voltaje C-
   A:\t\t"<< "[" << index << "] : " << a << std::endl;
214.                              break;
215.                             }
216.                     case 26:
217.                             {
218.                              std::cout <<"Numerador
   CT:\t\t\t"<< "[" << index << "] : " << a << std::endl;
219.                              break;
220.                             }
221.                     case 27:
222.                             {
223.                              std::cout <<"Multiplicador
   CT:\t\t"<< "[" << index << "] : " << a << std::endl;
224.                              break;
225.                             }
226.                     case 28:
227.                             {
228.                              std::cout <<"Denominador
   CT:\t\t\t"<< "[" << index << "] : " << a << std::endl;
229.                              break;
230.                             }
231.                     case 29:
232.                             {
233.                              std::cout <<"Numerador
   PT:\t\t\t"<< "[" << index << "] : " << a << std::endl;
234.                              break;
235.                             }
236.                     case 30:
237.                             {
238.                              std::cout <<"Multiplicador
   TP:\t\t"<< "[" << index << "] : " << a << std::endl;
239.                              break;
240.                             }
241.                     case 31:
242.                             {
243.                              std::cout <<"Denominador PT
   :\t\t"<< "[" << index << "] : " << a << std::endl;
244.                              break;
245.                             }
246.                     case 32:
247.                             {
248.                              a = (a*10)/32768;
249.                             std::cout <<"Corriente de
   neutro:\t\t"<< "[" << index << "] : " << a <<"\n"<< std::endl;
250.                              break;
251.                             }
252.                     }
253.             }
254.             template <class T>
```

```
255.             static std::string ValueToString(const T& meas)
256.             {
257.                     std::ostringstream oss;
258.                     oss << meas.value;
259.                     return oss.str();
260.             }
261.             static std::string GetTimeString(opendnp3::TimestampM
   ode tsmode)
262.             {
263.                     std::ostringstream oss;
264.                     switch (tsmode)
265.                     {
266.                     case(opendnp3::TimestampMode::SYNCHRONIZED) :
267.                             return "synchronized";
268.                             break;
269.                     case(opendnp3::TimestampMode::UNSYNCHRONIZED)
    :
270.                             oss << "unsynchronized";
271.                             break;
272.                     default:
273.                             oss << "no timestamp";
274.                             break;
275.                     }
276.                     return oss.str();
277.             }
278.
279.             static std::string ValueToString(const opendnp3::Doub
   leBitBinary& meas)
280.             {
281.                     return opendnp3::DoubleBitToString(meas.value
   );
282.             }
283.     };
284.     }
285.     #endif
```

Figura 36. Código para Impresión de valores reales (no escalados) en la terminal de linux.

Lo que ocurre en la plantilla de función, específicamente en la función Print, a partir de la línea 52 de la Figura 36, es que la interrogación se da el número total de veces que se le asigna al programa en cuestión de rango en el programa principal (main.cpp). En nuestro caso, hemos permitido que el rango vaya desde el punto 0 hasta el punto 32. Por lo cual se realizan 33 llamados a la función Print, porque cada uno se realiza por separado. Y en cada una de las llamadas el índice (index) aumentara en uno, por lo cual se imprimen todos los datos correctamente en la terminal.

3.7.2.2 Ajustes en código main.cpp para cambios de J_PrintingSOEhandler.

Junto a estas modificaciones, el archivo main.cpp (Codigo mostrado en la Figura 36) que es el archivo principal, debe ser modificado también y redireccionado ya no a

printingSOEhandler.h sino a J_printingSOEhandler.h.

Los cambios en las líneas correspondientes, se muestran en la Figura 37.

```
2. #include <asiodnp3/J_PrintingSOEHandler.h>

.
.
.

33. auto master = channel->AddMaster ("master",
J_PrintingSOEHandler::Create(),
asiodnp3::DefaultMasterApplication::Create(), stackConfig);
```

Figura 37. Configuraciones a realizar en archivo principal main.cpp.

Se debe incluir la librería J_printingSOEhandler.h en lugar de la librería printingSOEhandler.h (véase línea 2 de la Figura 30) y se debe redireccionar al maestro hacia J_printingSOEhandler.h (línea 33 de Figura 30).

3.7.2.3 Modificación de CMakeLists.txt para impresión analógica en terminal.

Debe recalcarse que CMake, compila en este caso los códigos correspondientes a Los ejemplos de master, Outstation y master-gprs, pero para nuestro trabajo, solo es necesario tener habilitado el apartado correspondiente a ”master”. Si se dejan habilitadas las otras características, el archivo CMakeLists.txt, ocurrirán errores, porque los demás ejemplos, están direccionados hacia printingSOEhandler.h y no hacia J_printingSOEhandler.h. A continuación en la Figura 38 se muestra la porción de código que se debe **suprimir** del archivo CMakeLists.txt para esta aplicación.

```
158.        # ----- master demo executable -----
159.        add_executable(master-gprs-demo ./cpp/examples/master-
   gprs/main.cpp)
160.        target_link_libraries (master-gprs-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
161.        set_target_properties(master-gprs-demo PROPERTIES FOLDER
   demos)
162.        # ----- outstation demo executable -----
163.        add_executable(outstation-demo
   ./cpp/examples/outstation/main.cpp)
164.        target_link_libraries (outstation-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
165.        set_target_properties(outstation-demo PROPERTIES FOLDER
   demos)
```

Figura 38. Datos a eliminar de CMakeLists.txt para el uso de los códigos J_printingSOEhandler.h y J_printingSOEhandler.cpp.

Al tener todos los cambios mencionados realizados, se procede a la compilación de la Biblioteca estando ubicados en la carpeta dnp3 desde la terminal.

Se sigue las instrucciones de compilación mencionadas en la sección 3.7. En caso de no existir errores el código se compilara correctamente y se procede a correr el ejecutable que tiene nombre de master-demo. El resultado de la interrogación, se muestra a continuación en la Figura 39.

```
ms(1537309071189) INFO    master - Begining task: Enable Unsolicited
ms(1537309071189) --AL->  master - C2 14 3C 02 06 3C 03 06 3C 04 06
ms(1537309071189) --AL->  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 2 FUNC: ENABLE_UNSOLICITED
ms(1537309071189) --AL->  master - 060,002 - Class Data - Class 1 - all objects
ms(1537309071189) --AL->  master - 060,003 - Class Data - Class 2 - all objects
ms(1537309071189) --AL->  master - 060,004 - Class Data - Class 3 - all objects
ms(1537309071235) <-AL--  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 2 FUNC: RESPONSE IIN: [0x00, 0x01]
ms(1537309071235) WARN    master - Task was explicitly rejected via response with error IIN bit(s): Enable
Unsolicited
ms(1537309071235) INFO    master - Begining task: Application Poll
ms(1537309071235) --AL->  master - C3 01 1E 04 01 00 00 20 00
ms(1537309071235) --AL->  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 3 FUNC: READ
ms(1537309071235) --AL->  master - 030,004 Analog Input - 16-bit Without Flag, 16-bit start stop [0, 32]
ms(1537309071360) <-AL--  master - FIR: 1 FIN: 1 CON: 0 UNS: 0 SEQ: 3 FUNC: RESPONSE IIN: [0x00, 0x00]
ms(1537309071360) <-AL--  master - 030,004 Analog Input - 16-bit Without Flag, 8-bit start stop [0, 32]

Salud del medidor:                [0] : 0
Voltaje A:                        [1] : 119.238
Voltaje B:                        [2] : 0
Voltaje C:                        [3] : 0
Voltaje A-B:                      [4] : 119.238
Voltaje B-C:                      [5] : 0
Voltaje C-A:                      [6] : 119.238
Corriente A:                      [7] : 0.0128174
Corriente B:                      [8] : 0
Corriente C:                      [9] : 0
Watts, 3ph total:                 [10] : 0
VARs, 3ph total:                  [11] : -1.37329
VAs, 3ph total:                   [12] : 1.37329
Factor de potencia 3ph total:     [13] : -0.013
Frecuencia:                       [14] : 59.96
Watts positivos 3ph:              [15] : 3001.19
VARs positivos 3ph:               [16] : 2817.99
Watts negativos 3ph:                    [17] : -871.353
VARs negativos 3ph:               [18] : -3154.59
VAs 3ph maximos:                  [19] : 3593.08
```

```
Angulo de fase. Corriente A:        [20] : 9.07
Angulo de fase. Corriente B:        [21] : 0
Angulo de fase. Corriente C:        [22] : 0
Angulo. Voltaje A-B:                [23] : -11.68
Angulo. Voltaje B-C:                [24] : 5.51
Angulo. Voltaje C-A:                [25] : 6
Numerador CT:                       [26] : 400
Multiplicador CT:                   [27] : 1
Denominador CT:                     [28] : 5
Numerador PT:                       [29] : 208
Multiplicador TP:                   [30] : 1
Denominador PT :                    [31] : 208
Corriente de neutro:                [32] : 0.0128174
channel state change: CLOSED
channel state change: SHUTDOWN
ms(1537309071975) INFO    manager - Exiting thread (0)
root@jose-Latitude-D510:/home/jose/Escritorio/prueba/dnp3#
```

Figura 39. Impresión de valor analógico de todos los puntos del mapa DNP del manual SHARK 200S en la Terminal de linux.

Debido a que en esta práctica el medidor no tiene carga conectada, algunos de los datos parecen carecer de sentido, sin embargo, hay datos correctos como el Voltaje A, la frecuencia entre otros.

3.7.3 Ejemplo 3. Colocar todos los puntos con valores analógicos en un archivo de texto.

Ya teniendo la capacidad de imprimir los datos en la terminal respectivamente con su valor analógico, se procede a modificar la plantilla de función Print de J_printingSOEhandler.h, para que en cada caso, abra un archivo de texto, e imprima dentro del archivo, los datos de cada punto en las 33 interrogaciones que se realizan.

A continuación, en la Figura 40 se muestra el código correspondiente solo a la plantilla de función Print en la que los mismos datos que se impriman en la terminal, se guardaran en el archivo de texto.

```
52.     template <class T>
53.             static void Print(const opendnp3::HeaderInfo& info, c
   onst T& value, uint16_t index)
54.             {
55.                     std::ofstream archivo;
56.                     std::string jose = ValueToString(value);
57.                     float a;
58.                     std::istringstream(jose) >> a;
59.                     switch(index)
60.                     {
```

```
61.                     case 0:
62.                             {
63.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::out);
64.                             std::cout <<"\n"<<"Salud del
   medidor:\t\t"<< "[" << index << "] : " << a << std::endl;
65.                             archivo<<"\n"<<"Salud del
   medidor:\t\t"<< "[" << index << "] : " <<a<<std::endl;
66.                             break;
67.                             }
68.                     case 1:
69.                             {
70.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
71.                             a = (a*150)/32768;
72.                             std::cout <<"Voltaje
   A:\t\t\t"<< "[" << index << "] : " << a << std::endl;
73.                             archivo<<"Voltaje
   A:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
74.                             break;
75.                             }
76.                     case 2:
77.                             {
78.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
79.                             a = (a*150)/32768;
80.                             std::cout <<"Voltaje
   B:\t\t\t"<< "[" << index << "] : " << a << std::endl;
81.                             archivo<<"Voltaje
   B:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
82.                             break;
83.                             }
84.                     case 3:
85.                             {
86.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
87.                             a = (a*150)/32768;
88.                             std::cout <<"Voltaje
   C:\t\t\t"<< "[" << index << "] : " << a << std::endl;
89.                             archivo<<"Voltaje
   C:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
90.                             break;
91.                             }
92.                     case 4:
93.                             {
94.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
95.                             a = (a*300)/32768;
96.                             std::cout <<"Voltaje A-
   B:\t\t\t"<< "[" << index << "] : " << a << std::endl;
97.                             archivo<<"Voltaje A-
   B:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
98.                             break;
```

```
99.                                 }
100.                        case 5:
101.                                {
102.                                archivo.open("/home/jose/Escritorio/a
    rchivito.txt",std::fstream::app);
103.                                a = (a*300)/32768;
104.                                std::cout <<"Voltaje B-
    C:\t\t\t"<< "[" << index << "] : " << a << std::endl;
105.                                archivo<<"Voltaje B-
    C:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
106.                                break;
107.                                }
108.                        case 6:
109.                                {
110.                                archivo.open("/home/jose/Escritorio/a
    rchivito.txt",std::fstream::app);
111.                                a = (a*300)/32768;
112.                                std::cout <<"Voltaje C-
    A:\t\t\t"<< "[" << index << "] : " << a << std::endl;
113.                                archivo<<"Voltaje C-
    A:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
114.                                break;
115.                                }
116.                        case 7:
117.                                {
118.                                archivo.open("/home/jose/Escritorio/a
    rchivito.txt",std::fstream::app);
119.                                a = (a*10)/32768;
120.                                std::cout <<"Corriente
    A:\t\t\t"<< "[" << index << "] : " << a << std::endl;
121.                                archivo<<"Corriente
    A:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
122.                                break;
123.                                }
124.                        case 8:
125.                                {
126.                                archivo.open("/home/jose/Escritorio/a
    rchivito.txt",std::fstream::app);
127.                                a = (a*10)/32768;
128.                                std::cout <<"Corriente
    B:\t\t\t"<< "[" << index << "] : " << a << std::endl;
129.                                archivo<<"Corriente
    B:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
130.                                break;
131.                                }
132.                        case 9:
133.                                {
134.                                archivo.open("/home/jose/Escritorio/a
    rchivito.txt",std::fstream::app);
135.                                a = (a*10)/32768;
136.                                std::cout <<"Corriente
    C:\t\t\t"<< "[" << index << "] : " << a << std::endl;
```

```
137.                                archivo<<"Corriente
   C:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
138.                                break;
139.                                }
140.                        case 10:
141.                                {
142.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
143.                                a = (a*4500)/32768;
144.                                std::cout <<"Watts, 3ph
   total:\t\t"<< "[" << index << "] : " << a << std::endl;
145.                                archivo<<"Watts, 3ph
   total:\t\t"<< "[" << index << "] : " <<a<<std::endl;
146.                                break;
147.                                }
148.                        case 11:
149.                                {
150.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
151.                                a = (a*4500)/32768;
152.                                std::cout <<"VARs, 3ph
   total:\t\t"<< "[" << index << "] : " << a << std::endl;
153.                                archivo<<"VARs, 3ph
   total:\t\t"<< "[" << index << "] : " <<a<<std::endl;
154.                                break;
155.                                }
156.                        case 12:
157.                                {
158.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
159.                                a = (a*4500)/32768;
160.                                std::cout <<"VAs, 3ph
   total:\t\t\t"<< "[" << index << "] : " << a << std::endl;
161.                                archivo<<"VAs, 3ph
   total:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
162.                                break;
163.                                }
164.                        case 13:
165.                                {
166.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
167.                                a = a*0.001;
168.                                std::cout <<"Factor de potencia 3ph
   total:\t"<< "[" << index << "] : " << a << std::endl;
169.                                archivo<<"Factor de potencia 3ph
   total:\t"<< "[" << index << "] : " <<a<<std::endl;
170.                                break;
171.                                }
172.                        case 14:
173.                                {
174.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
175.                                a = a*0.01;
```

```
176.                                std::cout <<"Frecuencia:\t\t\t"<< "["
    << index << "] : " << a << std::endl;
177.                                archivo<<"Frecuencia:\t\t\t"<< "[" <<
    index << "] : " <<a<<std::endl;
178.                                break;
179.                                }
180.                        case 15:
181.                                {
182.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
183.                                a = (a*4500)/32768;
184.                                std::cout <<"Watts positivos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
185.                                archivo<<"Watts positivos
   3ph:\t\t"<< "[" << index << "] : " <<a<<std::endl;
186.                                break;
187.                                }
188.                        case 16:
189.                                {
190.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
191.                                a = (a*4500)/32768;
192.                                std::cout <<"VARs positivos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
193.                                archivo<<"VARs positivos
   3ph:\t\t"<< "[" << index << "] : " <<a<<std::endl;
194.                                break;
195.                                }
196.                        case 17:
197.                                {
198.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
199.                                a = (a*4500)/32768;
200.                                std::cout <<"Watts negativos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
201.                                archivo<<"Watts negativos
   3ph:\t\t"<< "[" << index << "] : " <<a<<std::endl;
202.                                break;
203.                                }
204.                        case 18:
205.                                {
206.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
207.                                a = (a*4500)/32768;
208.                                std::cout <<"VARs negativos
   3ph:\t\t"<< "[" << index << "] : " << a << std::endl;
209.                                archivo<<"VARs negativos
   3ph:\t\t"<< "[" << index << "] : " <<a<<std::endl;
210.                                break;
211.                                }
212.                        case 19:
213.                                {
```

```
214.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
215.                                a = (a*4500)/32768;
216.                                std::cout <<"VAs 3ph
   maximos:\t\t"<< "[" << index << "] : " << a << std::endl;
217.                                archivo<<"VAs 3ph
   maximos:\t\t"<< "[" << index << "] : " <<a<<std::endl;
218.                                break;
219.                                }
220.                        case 20:
221.                                {
222.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
223.                                a = a*0.01;
224.                                std::cout <<"Angulo de fase.
   Corriente A:\t"<< "[" << index << "] : " << a << std::endl;
225.                                archivo<<"Angulo de fase. Corriente
   A:\t"<< "[" << index << "] : " <<a<<std::endl;
226.                                break;
227.                                }
228.                        case 21:
229.                                {
230.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
231.                                a = a*0.01;
232.                                std::cout <<"Angulo de fase.
   Corriente B:\t"<< "[" << index << "] : " << a << std::endl;
233.                                archivo<<"Angulo de fase. Corriente
   B:\t"<< "[" << index << "] : " <<a<<std::endl;
234.                                break;
235.                                }
236.                        case 22:
237.                                {
238.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
239.                                a = a*0.01;
240.                                std::cout <<"Angulo de fase.
   Corriente C:\t"<< "[" << index << "] : " << a << std::endl;
241.                                archivo<<"Angulo de fase. Corriente
   C:\t"<< "[" << index << "] : " <<a<<std::endl;
242.                                break;
243.                                }
244.                        case 23:
245.                                {
246.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
247.                                a = a*0.01;
248.                                std::cout <<"Angulo. Voltaje A-
   B:\t\t"<< "[" << index << "] : " << a << std::endl;
249.                                archivo<<"Angulo. Voltaje A-
   B:\t\t"<< "[" << index << "] : " <<a<<std::endl;
250.                                break;
251.                                }
```

```
252.                     case 24:
253.                             {
254.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
255.                             a = a*0.01;
256.                             std::cout <<"Angulo. Voltaje B-
   C:\t\t"<< "[" << index << "] : " << a << std::endl;
257.                             archivo<<"Angulo. Voltaje B-
   C:\t\t"<< "[" << index << "] : " <<a<<std::endl;
258.                             break;
259.                             }
260.                     case 25:
261.                             {
262.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
263.                             a = a*0.01;
264.                             std::cout <<"Angulo. Voltaje C-
   A:\t\t"<< "[" << index << "] : " << a << std::endl;
265.                             archivo<<"Angulo. Voltaje C-
   A:\t\t"<< "[" << index << "] : " <<a<<std::endl;
266.                             break;
267.                             }
268.                     case 26:
269.                             {
270.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
271.                             std::cout <<"Numerador
   CT:\t\t\t"<< "[" << index << "] : " << a << std::endl;
272.                             archivo<<"Numerador
   CT:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
273.                             break;
274.                             }
275.                     case 27:
276.                             {
277.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
278.                             std::cout <<"Multiplicador
   CT:\t\t"<< "[" << index << "] : " << a << std::endl;
279.                             archivo<<"Multiplicador
   CT:\t\t"<< "[" << index << "] : " <<a<<std::endl;
280.                             break;
281.                             }
282.                     case 28:
283.                             {
284.                             archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
285.                             std::cout <<"Denominador
   CT:\t\t\t"<< "[" << index << "] : " << a << std::endl;
286.                             archivo<<"Denominador
   CT:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
287.                             break;
288.                             }
289.                     case 29:
```

```
290.                                {
291.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
292.                                std::cout <<"Numerador
   PT:\t\t\t"<< "[" << index << "] : " << a << std::endl;
293.                                archivo<<"Numerador
   PT:\t\t\t"<< "[" << index << "] : " <<a<<std::endl;
294.                                break;
295.                                }
296.                        case 30:
297.                                {
298.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
299.                                std::cout <<"Multiplicador
   TP:\t\t"<< "[" << index << "] : " << a << std::endl;
300.                                archivo<<"Multiplicador
   TP:\t\t"<< "[" << index << "] : " <<a<<std::endl;
301.                                break;
302.                                }
303.                        case 31:
304.                                {
305.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
306.                                std::cout <<"Denominador PT
   :\t\t"<< "[" << index << "] : " << a << std::endl;
307.                                archivo<<"Denominador PT
   :\t\t"<< "[" << index << "] : " <<a<<std::endl;
308.                                break;
309.                                }
310.                        case 32:
311.                                {
312.                                archivo.open("/home/jose/Escritorio/a
   rchivito.txt",std::fstream::app);
313.                                a = (a*10)/32768;
314.                                std::cout <<"Corriente de
   neutro:\t\t"<< "[" << index << "] : " << a <<"\n"<< std::endl;
315.                                archivo<<"Corriente de
   neutro:\t\t"<< "[" << index << "] : " <<a<<std::endl;
316.                                break;
317.                                }
318.                        }
319.             }
```

Figura 40. Código para Impresión de valor real (no escalado) de todos los puntos del mapa DNP del manual SHARK 200S en la Terminal de linux y en un archivo de texto.

Para que los datos puedan imprimirse uno tras otro en el archivo de texto fue necesario en cada caso, agregarle la siguiente característica a la apertura del archivo.

```
std::fstream::app
```

Esa instrucción permite imprimir datos, agregandolos a un archivo ya existente. Y en caso de no exixtir el archivo, lo crea. Sin embargo, podria ocurrir un problema. Si el programa se corre en mas de una ocasión, los datos de todas las interrogaciones se montarian en el mismo archivo. Los datos se irian almacenando interrogacion tras interrogacion y el archivo se iria llenando y haciendose mas grande cada vez.
La solucion a este problema es. Escribir la siguiente linea en la apertura del primer caso (caso en el que index vale 0).

std::fstream::out

Esta instrucción crea un archivo. Y si esta ya creado un archivo con el mismo nombre lo sustituye. Y por ende todos los datos correspondientes que estan dentro del archivo se pierden.

Siendo asi, nada mas se mostraran especificamente los datos consultados en el momento y los de la anterior interrogacion se borraran del archivo de texto. Al realizar los cambios correspondientes y compilar el codigo, si no hay errores en el proceso, se obtienen los resultados, tanto en la terminal, como en el archivo de texto como se puede observar en la Figura 41.

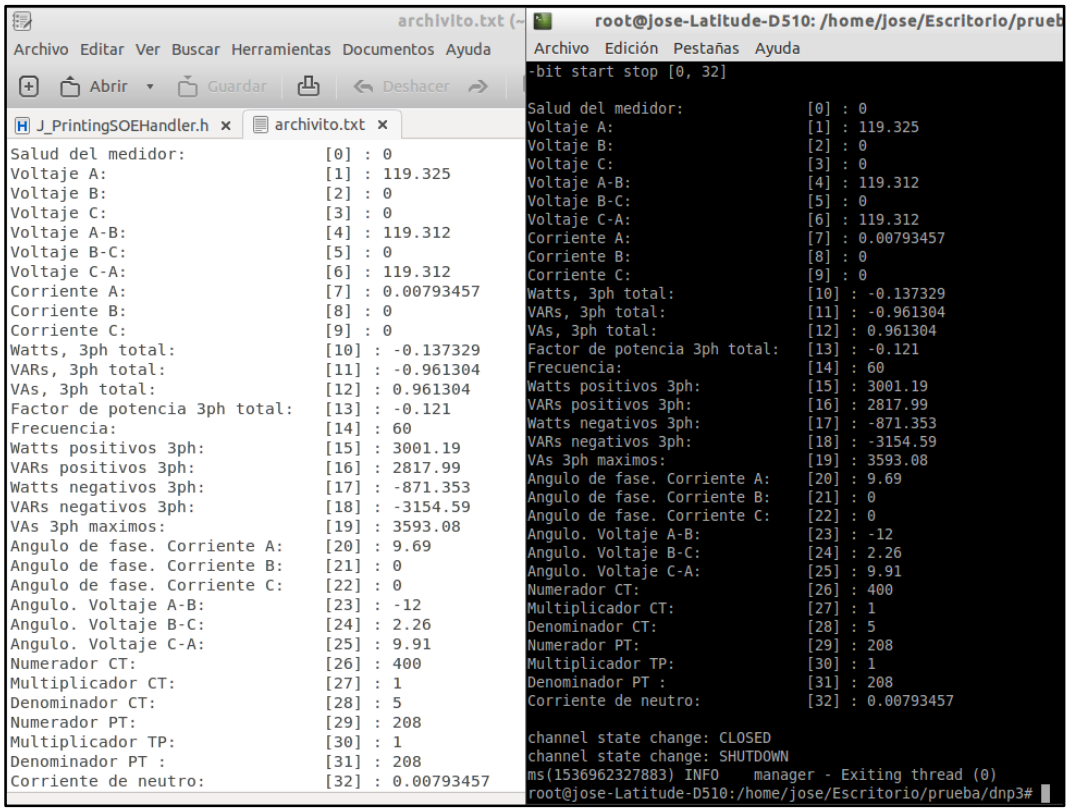

Figura 41. Obtención correcta de datos tanto en la terminal de Linux como en el archivo de texto.

3.7.4 Ejemplo 4. Ingreso de datos a tabla MySQL.

3.7.4.1 creacion del servidor e instalacion de MySQL.

Algo importante hasta este punto, es que ya se tiene disposicion de los datos ya sea en variables y/o en archivos de texto. Esos datos pueden ser manipulados para su posterior procesamiento.

En este ejemplo, se almacenara la unica variable de interes en cuanto a las pruebas que se realizaron en el laboratorio. El dato de interes, es el voltaje en la fase A.

Para ingresar datos a una tabla MySQL, se debe realizar principalmente la instalacion de MySQL y el establecimiento del servidor Apache ya sea en la computadora, o en la Raspberry pi. En nuestro caso, para establecer el servidor, se ha realizado el procedimiento descrito especificamente en el anexo I de [11].

3.7.4.2 Definicion de un usuario y una contraseña en MySQL.

Posterior a la creacion del servidor, y la instalacion de MySQL, se debe ingresar a MySQL con la forma por defecto. Esa forma de ingreso es la que se muestra acontinuacion.

```
root@jose-Latitude-D510:/home/jose# mysql -u root -p
```

Se pedira la contraseña de superusuario especificada al momento de la instalacion de MySQL y al escribirla se logra acceder a MySQL. En ese momento se debe proceder a crear un nuevo usuario en el cual se debe especificar un nombre de usuario, el numero de la IP del servidor Apache y una contraseña, como se muestra acontinuacion.

```
mysql>CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'password';
```

Con la linea anterior, se define el usuario (puede colocarse el nombre que se desee), en localhost, debe colocarse la IP correspondiente al servidor (generalmente 127.0.0.1), y en el ultimo apartado en lugar de la palabra password, se puede colocar la contraseña que se desee.

Aun al crear el usuario, no sera posible acceder a MySQL, debido a que a dicho usuario se le deben otorgar privilegios.Para este caso es recomendable otorgarle al usuario todos los privilegios. Eso se realiza con la siguiente linea .

```
mysql>GRANT ALL PRIVILEGES ON * . * TO 'usuario'@'localhost';
```

Donde debe cambiarse el apartado “usuario” por el nombre que se le ha dado al usuario creado y en “localhost” el numero de la IP del servidor. Por ultimo luego de haberle otorgado todos los privilegios, debemos escribir la sigiente linea para que surtan efecto los cambios en MySQL.

```
mysql>FLUSH PRIVILEGES;
```

Ahora al salir de MySQL, y volver a entrar, ya se puede acceder teniendo un usuario especifico y una contraseña. Este paso es necesario para poder identificar como se ingresara desde el programa escrito en c++. Si no se tiene un usuario y una contraseña, no es posible el ingreso.

Para ingresar de manera manual a MySQL con el usuario creado, debe escribirse la siguiente linea en la terminal de linux.

```
root@jose-Latitude-D510:/home/jose# mysql --user=   name  --password= pass
```

Donde “name” es el nombre que se le asigno al usuario que se ha creado y “pass” es la “contraseña” que el usuario asigno al usuario MySQL que ha creado.

3.8.4.3 instalacion de biblioteca libmysql.

Para poder realizar programas que corran correctamente en entorno Linux, se debe instalar las características libmysql y libmysql++. Para ello, se escribe la siguiente línea en la terminal de Linux.

```
root@jose-Latitude-D510:/home/jose# sudo apt-get install libmysql++ libmysql++-dev
```

Con esto ya se pueden compilar y correr programas utilizando estas bibliotecas pero, solo con el compilador g++. Para poder compilar con CMake, se debe realizar una configuracion en el archivo CMakelist.txt de la carpeta dnp3.

3.7.4.4 Modificacion de archivo CMakeLists.txt de carpeta dnp3.

En la porción correspondiente a DNP3_DEMO, el único apartado disponible debe ser el de la creación de ejecutable master-demo. Y para este fin, ese apartado debe modificarse a como se observa en la Figura 42.

```
153.      if(DNP3_DEMO)
154.      # ----- master demo executable -----
155.      include_directories(/usr/include/mysql++/)
156.      include_directories(/usr/include/mysql/)
157.      add_executable(master-demo ./cpp/examples/master/main.cpp)
158.      target_link_libraries (master-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
159.      target_link_libraries(master-demo mysqlpp)
160.      set_target_properties(master-demo PROPERTIES FOLDER demos)
```

Figura 42. Configuración de archivo CMakeLists.txt.

Lo que se hizo, fue incluir los directorios externos a la biblioteca OpenDNP3 con sus respectivas ubicaciones. En este caso se han incluido los directorios que contienen las librerias .h de la biblioteca mysql y mysql++. Realizada ya laconfiguracion en el archivo CMakeLists.txt, puede ahora armarse los codigos y estos compilaran de manera correcta con CMake.

3.7.4.5 Porcion de codigo J_PrintingSOEhandler.h agregada para MySQL.

El programa a realizar en c++ es una edicion del mismo que se ha mencionado en los ejemplos anteriores. Acontinuacion en la Figura 43, se muestra solo la plantilla de funcion correspondiente a las impresiones de toda la biblioteca (codigo J_PrintingSOEhandler.h) editada ya con el codigo de MySQL correspondiente.

```
52.             template <class T>
53.             static void Print(const opendnp3::HeaderInfo& info, c
   onst T& value, uint16_t index)
54.             {
55.                     std::string jose = ValueToString(value);
56.                     float a;
57.                     std::istringstream(jose) >> a;
58.                     switch(index)
59.                     {
60.                     case 1:
61.                        {
62.                        a=(150/32768)*a;
63.                        std::ostringstream ss;
64.                        ss << a;
65.                        std::string s(ss.str());
66.                        //procedimiento a MySQL
67.                        mysqlpp::Connection conn(false);
68.                        char server[] = "127.0.0.1";
69.                        char user[]   = "jose";  // O cualquier
   usuario
70.                        char pass[]   = "root";
71.                        if (conn.connect(NULL, server, user,
   pass))
72.                           {
73.                            std::string consulta="SHOW DATABASES";
74.                            mysqlpp::Query query = conn.query(cons
   ulta);
75.                            std::cout << "Creando base de
   datos" << std::endl;
76.                            if (conn.create_db("pruebaDNP3"))
77.                                {
78.                                std::cout << "Seleccionando base
   de datos" << std::endl;
79.                                if (conn.select_db("pruebadnp3"))
80.                                   {
81.                                   std::cout << "Creando
   tabla" << std::endl;
82.                                   query.reset();
83.                                   query=conn.query("CREATE TABLE
   IF NOT EXISTS `poesiadb`.`test` (`id` bigint(20) NOT NULL
   AUTO_INCREMENT,`variable` varchar(100) NOT NULL,`valor` FLOAT NOT
   NULL,PRIMARY KEY (`id`)) ENGINE=MyISAM DEFAULT CHARSET=latin1
   AUTO_INCREMENT=1 ;");
84.                                         if (query.execute())
85.                                            {
```

```
86.                                             query.reset();
87.                                             query << "INSERT INTO
   `test` VALUES (NULL, 'Voltaje_A', '"+s+"');";
88.                                             if(!query.execute())
89.                                             query.reset();}
90.                                         }
91.                                     }
92.                                 }
93.                             conn.disconnect();
94.                             break;
95.                             }
96.                     }
97.             }
```

Figura 43. Código en c++ para ingresar datos a MySQL.

Las lineas 54 - 57 de este programa, se hace la transformacion del valor de la variable “value” , a un valor flotante, y posteriormente pasado a la variable “a”. Las lineas 58 - 61 establecen un caso (case). En el cual se detecta especificamente que la variable “index” tenga un valor de 1. Por que es en ese momento en que se estara obteniendo el dato de voltaje en la fase A.

Las lineas 62 y 65 hacen la conversion del valor flotante mapeado desde 0 hasta 32768 al valor analogico real de voltaje y se vuelve a convertir el valor en un “string”, por que en la instrucción de la linea 87, solo se permite la impresión de cadenas de caracteres o “strings” y no de numeros flotantes.

Las lineas 67 y 70 especifican los datos del usuario que se creo en la sección 3.7.4.2. los cuales serviran para que el programa pueda acceder a MySQL y realizar el procedimiento respectivo.

Se tiene la opcion de hacer manualmente una base de datos, e ingresarle a ella una tabla, y posteriormente en el programa solo conectarse a la tabla y enviar los datos. Pero en este caso se ha decidido realizar todo el procedimiento desde el mismo codigo.

Las lineas 76 y 80 se crean la base de datos pruebaDNP3y especificamente en la linea 79 se selecciona precisamente esa base de datos para operar en ella. Las linea 81 - 83 crean la tabla “datos”, en la que se especifica que hay un campo llamado “id”, el cual lleva un conteo automatico de los datos mediante estos se van agregando a la tabla. El segundo campo especificado es “variable”, en el que se imprime el nombre del dato interrogado. En este caso, es el de “Voltaje_A”, y el ultimo campo permite la entrada de un flotante y se le coloco el nombre “valor”.

Luego de definir los campos de la tabla, entre las lineas 84 y 89, se imprime finalmente el valor con la instruccion “INSERT INTO” de MySQL. Posteriormente, se procede a compilar la biblioteca, se corre el archivo master-demo, y en la terminal no aparece nada a parte de la impresión de los mensajes especificados en el programa. Sin embargo, al ingresar a MySQL, elegir la base de datos realizada y observar los valores que han ingresado a la tabla, observamos en este caso lo siguiente.

```
mysql> use pruebaDNP3;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from datos;
+------+--------------+------------+
| id   | variable     | valor      |
+----+---------------+------------+
|  1   | Voltaje_A  | 118.425 |
+----+--------------+--------------+
1 row in set (0.00 sec)

mysql>
```

Figura 44. Obtención correcta del dato de Voltaje A en MySQL.

3.7.5 Analisis de puerto serie mediante Wireshark.

Acontinuacion se muestra el resultado de una prueba de Wireshark obtenida mediante se realizaba una simulacion como la descrita en la sección 3.5. En este caso Wireshark no detecta la operación como trama DNP3, debido a que fue echa por medio del puerto serie con el convertidor USB-RS485.

Los mensajes en los que se detalla el “host” son los enviados desde el maestro hacia el esclavo entre estos mensajes se pueden hayar tanto solicitudes, confirmaciones como tambien “ACKs”, y los enviados desde el dispositivo “4.2.0”, son enviados como respuestas desde la estacion remota. Entre estos hay respuestas y tambien “ACKs” de confirmacion.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	host	4.2.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
2	0.000000	4.2.0	host	USB	67	GET DESCRIPTOR Response CONFIGURATION
3	0.000000	4.2.0	host	USB	28	GET DESCRIPTOR Status
4	0.000000	host	4.2.0	USB	36	SET CONFIGURATION Request
5	0.015600	4.2.0	host	USB	28	SET CONFIGURATION Status
6	0.015600	host	4.2.0	USB	36	URB_CONTROL in
7	0.015600	4.2.0	host	USB	30	URB_CONTROL in
8	0.015600	4.2.0	host	USB	28	Unknown type 5f Status
9	0.015600	host	4.2.0	USB	36	URB_CONTROL out
10	0.015600	4.2.0	host	USB	28	Unknown type a1 Status
11	0.015600	host	4.2.0	USB	36	URB_CONTROL out
12	0.015600	4.2.0	host	USB	28	Unknown type 9a Status
13	0.015600	host	4.2.0	USB	36	URB_CONTROL out

14 0.015600	4.2.0	host	USB	28	Unknown type 9a Status
15 0.015600	host	4.2.0	USB	36	URB_CONTROL in
16 0.031200	4.2.0	host	USB	30	URB_CONTROL in
17 0.031200	4.2.0	host	USB	28	Unknown type 95 Status
18 0.031200	host	4.2.0	USB	36	URB_CONTROL in
19 0.031200	4.2.0	host	USB	30	URB_CONTROL in
20 0.031200	4.2.0	host	USB	28	Unknown type 95 Status
21 0.031200	host	4.2.0	USB	36	URB_CONTROL out
22 0.031200	4.2.0	host	USB	28	Unknown type 9a Status
23 0.046800	host	4.2.0	USB	36	URB_CONTROL out
24 0.046800	4.2.0	host	USB	28	Unknown type a1 Status
25 0.046800	host	4.2.0	USB	36	URB_CONTROL out
26 0.046800	4.2.0	host	USB	28	Unknown type 9a Status
27 0.046800	host	4.2.0	USB	36	URB_CONTROL out
28 0.046800	4.2.0	host	USB	28	Unknown type a4 Status
29 0.046800	host	4.2.0	USB	36	URB_CONTROL out
30 0.046800	4.2.0	host	USB	28	Unknown type a4 Status
31 0.046800	host	4.2.0	USB	36	URB_CONTROL in
32 0.062400	4.2.0	host	USB	30	URB_CONTROL in
33 0.062400	4.2.0	host	USB	28	Unknown type 95 Status
34 0.062400	host	4.2.0	USB	36	URB_CONTROL out
35 0.062400	4.2.0	host	USB	28	Unknown type 9a Status
36 0.062400	host	4.2.0	USB	36	URB_CONTROL out
37 0.062400	4.2.0	host	USB	28	Unknown type a4 Status
38 0.062400	host	4.2.0	USB	36	URB_CONTROL out
39 0.062400	4.2.0	host	USB	28	Unknown type a4 Status
40 0.062400	host	4.2.0	USB	36	URB_CONTROL in
41 0.062400	4.2.0	host	USB	30	URB_CONTROL in
42 0.062400	4.2.0	host	USB	28	Unknown type 95 Status
43 0.062400	host	4.2.0	USB	36	URB_CONTROL out
44 0.078000	4.2.0	host	USB	28	Unknown type 9a Status
45 0.078000	host	4.2.0	USB	36	URB_CONTROL out
46 0.078000	4.2.0	host	USB	28	Unknown type a4 Status
47 0.278003	host	4.2.0	USB	36	URB_CONTROL in
48 0.282003	4.2.0	host	USB	30	URB_CONTROL in
49 0.282003	4.2.0	host	USB	28	Unknown type 95 Status
50 0.282003	host	4.2.0	USB	36	URB_CONTROL out
51 0.285003	4.2.0	host	USB	28	Unknown type 9a Status
52 0.285003	host	4.2.0	USB	36	URB_CONTROL out
53 0.288003	4.2.0	host	USB	28	Unknown type 9a Status
54 0.288003	host	4.2.0	USB	36	URB_CONTROL out

55 0.291003	4.2.0	host	USB	28	Unknown type 9a Status
56 0.291003	host	4.2.0	USB	36	URB_CONTROL out
57 0.294003	4.2.0	host	USB	28	Unknown type 9a Status
58 0.294003	host	4.2.0	USB	36	URB_CONTROL out
59 0.297004	4.2.0	host	USB	28	Unknown type a4 Status
60 0.498014	host	4.2.0	USB	36	URB_CONTROL out
61 0.498014	4.2.0	host	USB	28	Unknown type a4 Status
62 0.698016	host	4.2.0	USB	36	URB_CONTROL out
63 0.701017	4.2.0	host	USB	28	Unknown type a4 Status
64 0.942030	host	4.2.0	USB	36	URB_CONTROL out
65 0.945031	4.2.0	host	USB	28	Unknown type a4 Status
66 0.945031	host	4.2.0	USB	36	URB_CONTROL out
67 0.948031	4.2.0	host	USB	28	Unknown type a4 Status
68 1.148042	host	4.2.0	USB	36	URB_CONTROL out
69 1.151042	4.2.0	host	USB	28	Unknown type a4 Status
70 1.151042	host	4.2.0	USB	36	URB_CONTROL out
71 1.154043	4.2.0	host	USB	28	Unknown type a4 Status
72 1.395050	host	4.2.0	USB	36	URB_CONTROL out
73 1.398050	4.2.0	host	USB	28	Unknown type a4 Status
74 1.613054	host	4.2.0	USB	36	URB_CONTROL in
75 1.613054	4.2.0	host	USB	30	URB_CONTROL in
76 1.613054	4.2.0	host	USB	28	Unknown type 95 Status
77 1.613054	host	4.2.0	USB	36	URB_CONTROL out
78 1.613054	4.2.0	host	USB	28	Unknown type 9a Status
79 1.613054	host	4.2.0	USB	36	URB_CONTROL out
80 1.613054	4.2.0	host	USB	28	Unknown type a4 Status
81 1.815854	host	4.2.0	USB	36	URB_CONTROL out
82 1.815854	4.2.0	host	USB	28	Unknown type a4 Status
83 2.018655	host	4.2.0	USB	36	URB_CONTROL out
84 2.018655	4.2.0	host	USB	28	Unknown type a4 Status
85 2.268255	host	4.2.0	USB	36	URB_CONTROL out
86 2.268255	4.2.0	host	USB	28	Unknown type a4 Status
87 2.268255	host	4.2.0	USB	36	URB_CONTROL out
88 2.268255	4.2.0	host	USB	28	Unknown type a4 Status
89 2.471056	host	4.2.0	USB	36	URB_CONTROL out
90 2.471056	4.2.0	host	USB	28	Unknown type a4 Status
91 2.471056	host	4.2.0	USB	36	URB_CONTROL out
92 2.471056	4.2.0	host	USB	28	Unknown type a4 Status
93 2.720656	host	4.2.0	USB	36	URB_CONTROL out
94 2.720656	4.2.0	host	USB	28	Unknown type a4 Status
95 2.923456	host	4.2.0	USB	36	URB_CONTROL in

96 2.923456	4.2.0	host	USB	30	URB_CONTROL in
97 2.923456	4.2.0	host	USB	28	Unknown type 95 Status
98 2.923456	host	4.2.0	USB	36	URB_CONTROL out
99 2.923456	4.2.0	host	USB	28	Unknown type 9a Status
100 2.923456	host	4.2.0	USB	36	URB_CONTROL out
101 2.923456	4.2.0	host	USB	28	Unknown type a4 Status
102 3.126257	host	4.2.0	USB	36	URB_CONTROL out
103 3.126257	4.2.0	host	USB	28	Unknown type a4 Status
104 3.329057	host	4.2.0	USB	36	URB_CONTROL out
105 3.329057	4.2.0	host	USB	28	Unknown type a4 Status
106 3.578657	host	4.2.0	USB	36	URB_CONTROL out
107 3.578657	4.2.0	host	USB	28	Unknown type a4 Status
108 3.578657	host	4.2.0	USB	36	URB_CONTROL out
109 3.578657	4.2.0	host	USB	28	Unknown type a4 Status
110 3.781458	host	4.2.0	USB	36	URB_CONTROL out
111 3.781458	4.2.0	host	USB	28	Unknown type a4 Status
112 3.781458	host	4.2.0	USB	36	URB_CONTROL out
113 3.781458	4.2.0	host	USB	28	Unknown type a4 Status
114 4.031058	host	4.2.0	USB	36	URB_CONTROL in
115 4.031058	4.2.0	host	USB	30	URB_CONTROL in
116 4.031058	4.2.0	host	USB	28	Unknown type 95 Status
117 4.031058	host	4.2.0	USB	36	URB_CONTROL out
118 4.031058	4.2.0	host	USB	28	Unknown type 9a Status
119 4.031058	host	4.2.0	USB	36	URB_CONTROL in
120 4.031058	4.2.0	host	USB	30	URB_CONTROL in
121 4.031058	4.2.0	host	USB	28	Unknown type 95 Status
122 4.031058	host	4.2.0	USB	36	URB_CONTROL out
123 4.031058	4.2.0	host	USB	28	Unknown type a4 Status

Figura 45. Registro de mensajes obtenidos en WIRESHARK.

3.8 Similitudes de las practicas realizadas en Lubuntu 14.04 y Raspberry pi.

Las arquitecturas de ambas computadoras son diferentes, pero para efectos de este trabajo de graduacion, se realizaron pruebas simultaneamente en ambos dispositivos. La raspberry pi utilizada fue Raspberry pi 3 modelo B con distribucion Raspbian Jessie. Y no hubo diferencias significativas en la realizacion de las pruebas de codigo correspondientes tanto en la distribucion Lubuntu 14.04 como en Raspbian Jessie. Por lo cual puede aplicarse cualquier procedimiento para cualquiera de las dos distribuciones en cuestion y de buena manera, en ambas distribuciones se obtendran los resultados esperados.

4. Conclusiones y líneas futuras

4.1 Conclusiones

- A pesar de que las arquitecturas de las dos computadoras utilizadas. Una con sistema operativo Lubuntu 14.04 y la raspberry pi con Raspbian Jessie, son diferentes, el procedimiento para la instalación de las características y ejecución de CMake y Opendnp3 es el mismo.

- El software *Comunicator Ext*, tiene la capacidad de mostrar los datos de medición mediante los protocolos Modbus RTU y TCP, pero para poder acceder al protocolo DNP3 debe pagarse por la versión profesional del software. Es necesario agregar que los datos no pueden ser manipulados para almacenarse en una computadora y formar una red. Por otra parte, El simulador de OpenDNP3 muestra una sola trama de datos, pero de una manera no amigable con el usuario, y de igual manera que con *Comunicator Ext*, los datos no pueden ser transferidos a ningún lugar para darles un respectivo procesamiento. Por lo cual se han utilizado bibliotecas para los diferentes protocolos con los cuales se realizaron los códigos mostrados en este trabajo, y los resultados son favorables en cuando a la posibilidad del posterior procesamiento de datos.

- Se ha realizado compilación común en gcc para los códigos hechos en LibModbus y con CMake en OpenDNP3. Los ejecutables se generan de manera correcta dado que al correrse se obtienen los resultados esperados sin errores intermedios debidos al desarrollo de las bibliotecas.

- Los medidores de energía no tienen capacidad de almacenamiento de datos, sin embargo, utilizando la Raspberry pi para el desarrollo de este proyecto se puede dotar de la capacidad de memoria a un medidor, del tamaño que se desee, dependiendo del tamaño del micro SD insertada en la Raspberry. En el caso de este proyecto se trabajó con un medidor SHARK200S y una raspberry pi 3 modelos B con una micro SD de 16 GB.

- Se realizó correctamente la compilación en CMake de programas realizados con una combinación entre las librerías de OpenDNP3 y MySQL para lenguaje c++ y se logró introducir datos de voltaje en la fase A de manera exitosa en una tabla MySQL.

4.2 Líneas Futuras

- Incorporar más medidores de energía para las subestaciones no incluidas en la red. Medidores que cuenten con el protocolos de comunicación Modbus/TCP, Modbus RTU, DNP3 TCP y DNP3 RTU podrían ahora incorporarse al sistema con facilidad.

- Si se compran medidores con protocolo DNP3 TCP, podrá aplicarse el mismo sistema con el que cuenta la red de medidores pero la interrogación debe hacerse desde el servidor (Raspberry pi). Es necesario que un futuro trabajo de graduación, se base en la realización de compilación cruzada en routers MESH01 con la biblioteca OpenDNP3 para que la interrogación en ese caso pueda realizarse desde el router, y posteriormente enviarse al servidor.

- En caso de adquirirse medidores con protocolos Modbus y/o DNP3 RTU, el modelo de conexión que podría ser funcional, puede ser. Conectar desde el medidor, mediante el convertidor USB-RS485 hasta la Raspberry pi, y desde la Raspberry pi mediante el puerto Ethernet, conectar un router MESH01 para poder realizar la conexión de manera inalámbrica, manteniendo así la estructura de cómo se maneja la red de medidores hasta hoy.

- Investigar acerca de la board UniPi. La cual es un módulo de relés con entradas y salidas analógicas/digitales para la automatización del hogar. Ésta, al contar con un protocolo de comunicación industrial (Preferiblemente DNP3) podría llegar a ser una RTU de bajo costo. Lo interesante del caso, es que al combinar ambas ideas, RTU de bajo costo más un sistema SCADA en Raspberry Pi (O micro computadora similar) conectados a través del protocolo DNP3 podría tener aplicaciones no solo para empresas de distribución de energía eléctrica, sino también para el control de procesos industriales en empresa pequeñas.

ANEXOS

Anexo A. Instalación de CMake.

A.1 Instalación de características previas a CMake.

CMake es una herramienta que permite una compilación de código personalizada y por ende la creación de ejecutables. La compilación es realizada por medio de ficheros de configuración que en este caso son llamados CMakeLists.txt. Para realizar la interrogación, debido a que la biblioteca de opendnp3 tiene un diseño específico para compilación con CMake, es necesario el uso de esta herramienta. Por lo cual, en caso de que el equipo a utilizar no cuente con dicha herramienta debe instalarse tanto las características previas necesarias como la herramienta CMake en sí.

Es importante recalcar que las pruebas realizadas en el laboratorio, se hicieron en una PC con distribución Linux, con Lubuntu 14.04, y también con una Raspberry Pi 3 Modelo B y con distribución Raspbian Jessie. A continuación se muestran los pasos que dieron lugar a la instalación de los componentes necesarios para la correcta realización de la compilación de programas de la librería OpenDNP3.

Como primer paso Para la instalación Escribimos las líneas de actualización correspondientes del sistema.

```
root@jose-Latitude-D510:/home/jose# sudo apt-get update
root@jose-Latitude-D510:/home/jose# sudo apt-get upgrade
```

Con los pasos anteriores el equipo se encuentra preparado para que en él, se realice las instalaciones. Primero se debe instalar el compilador necesario para el correcto funcionamiento de CMake. Para la versión de CMake que instalaremos, es necesario a la vez, tener instalada la versión 4.9 de gcc y g++. Si estas herramientas no están instaladas ya en el equipo, para poder hacerlo, Primero elusuario debe loguearse como root.

```
root@jose-Latitude-D510:/home/jose# sudo su
```

Se pedirá la contraseña de súper usuario para poder utilizar el equipo en modo root. Escribimos la contraseña. Posteriormente seguimos los siguientes pasos para instalar características importantes necesarias y agregar también un repositorio que es útil para la compilación.

```
root@jose-Latitude-D510:/home/jose# apt- get install build-essential
root@jose-Latitude-D510:/home/jose# add-apt-repository ppa:ubuntu-toolchain-r/test
```

Por ultimo hacemos la instalación de gcc y g++ en su versión 4.9.

```
root@jose-Latitude-D510:/home/jose# apt-get install gcc-4.9 g++4.9 cpp-4.9
```

Al completarse la instalación podemos comprobar con la siguiente línea la versión instalada en nuestro equipo.

```
root@jose-Latitude-D510:/home/jose# g++ -- versión
```

En caso de aun no haberse actualizado a la nueva versión (versión 4.9), en segunda instancia, puede cerrarse la terminal y reabrirse y probar nuevamente con la línea anterior. En el último de los casos aún no se muestra la nueva versión, nada más debe reiniciarse el equipo y al encenderse nuevamente los cambios estarán ya aplicados. Es necesario recalcar que para la versión de CMake que se instalará es necesaria la versión 4.9 de gcc y g++.

A.2 Instalación y configuración de CMake.

Para los siguientes pasos, ya no es necesario ser Root. Por lo cual este procedimiento puede realizarse justo al momento de abrir la terminal. Estando ya instalado g++ y gcc 4.9, en Lubuntu 14.04 escribir la siguiente línea en la terminal.

```
root@jose-Latitude-D510:/home/jose# sudo apt-get install libasio-dev
```

Esta línea puede obviarse al realizarse el procedimiento en la Raspberri pi, porque no tiene ningún efecto, debido a que las arquitecturas de las PC convencionales y la raspberry pi son diferentes. Por otra parte las características necesarias que esa línea produce en Lubuntu 14.04 ya vienen incluidas en Raspbian, así que es indiferente si se escribe o no, al hacerse en Raspbian.

Ahora se iniciará con la instalación precisa de CMake. Habiendo ya realizado los pasos previos, esta instalación y configuración se realizara de manera correcta. Primero el usuario debe ubicarse en la dirección que desee donde puedan guardarse que se descargan y se descomprimen durante este proceso. Es recomendable ubicarse en un lugar no fácilmente accesible como “el escritorio”, porque de quedar allí, si por error se borra la carpeta que contiene a CMake, se deberá nuevamente realizar el proceso de instalación para recuperarla.

Algo recomendable puede ser, realizar el procedimiento en el caso de la Raspberry, en el directorio home/pi o su equivalente en lubuntu. Para proceder, en la terminal se debe escribir el siguiente comando para descargar un archivo Tar en el que se encuentran los archivos necesarios.

```
root@jose-Latitude-D510:/home/jose# wget https://cmake.org/files/v3.11/cmake-3.11.0-rc1.tar.gz
```

En el mismo lugar en el que estamos se debe descomprimir el archivo con la siguiente línea.

```
root@jose-Latitude-D510:/home/jose# Tar –zxvf cmake-3.11.0-rc1.tar.gz
```

Luego se debe acceder al directorio que apareció en la ubicación debido a la descompresión. Eso se hace con la siguiente línea.

```
root@jose-Latitude-D510:/home/jose# cd cmake-3.11-rc1
```

Estando dentro del directorio se escribie la siguiente línea que ejecuta el archivo que la que da lugar a la instalación y configuración.

```
root@jose-Latitude-D510:/home/jose# ./bootstrap
```

Esto tomara un tiempo un poco largo así que al realizarse se debe ser paciente y esperar a que el procedimiento termine sin interrupciones. De preferencia no permitiendo que la pantalla del computador se apague o que el equipo se suspenda o entre en Hibernación, dado que esa interrupción puede dar lugar a errores en el procedimiento. Luego de completarse este procedimiento escribimos en la línea de comandos:

```
root@jose-Latitude-D510:/home/jose# Make
```

Es necesario destacar que este procedimiento se tardara un tiempo considerable pero en este caso en la pantalla se va mostrando el proceso en líneas color verde y el porcentaje del progreso. Por ultimo para completar la instalación de CMake, escribimos lo siguiente:

```
root@jose-Latitude-D510:/home/jose# Make install
```

Y con esto el proceso de instalación de esta herramienta para compilar los programas “a nuestra manera”, ha terminado.

En caso de querer comprobar la versión instalada de CMake podemos escribir también en la terminal la siguiente línea de comandos.

```
root@jose-Latitude-D510:/home/jose# cmake --versión
```

Anexo B. Instalación y configuración de OpenDNP3

B.1 Descarga de Biblioteca OpenDNP3.

Se ha desarrollado una biblioteca que engloba el estándar IEEE 1815, y otras normativas europeas que tienen que ver con el protocolo DNP3. Actualmente, hasta la finalización del presente trabajo de graduación la versión de openDNP3 vigente es la 2.2.0. En caso de realizar una continuación a este proyecto en la posteridad, debe comprobarse cuales son los cambios en las versiones subsecuentes y tener en cuenta consideraciones al respecto. El sitio oficial de opendnp3 es [2].

En ese mismo sitio se puede encontrar una guía de uso que explica muchas cosas específicas de la aplicación de esta librería y un poco de la sintaxis utilizada en ella. La guía se encuentra en [3].

Y también de (sitio opendnp3) se puede llegar a la página de la documentación de la biblioteca, donde se dan breves explicaciones de las funciones, los archivos las clases, entre otras cosas y se muestran también los archivos.h correspondientes que configuran la librería. En este caso es importante recalcar que la extensión de la biblioteca en la que nosotros nos hemos basado está escrita en c++. La ubicación de la documentación para el enriquecimiento de conocimiento en openDNP3 se encuentra en [4].

Para poder descargar la carpeta con toda la biblioteca debemos ubicarnos en la página correspondiente de la guía denominada “CMake” que está ubicada en el apartado “Build”. Esa extensión específicamente se encuentra en [5].

En ese sitio específicamente se muestra la forma de descarga de la carpeta de la biblioteca. Para eso, debemos escribir en la terminal de Linux (ya sea en Lubuntu o Raspbian), la siguiente línea.

```
root@jose-Latitude-D510:/home/jose# git clone –recursive https://github.com/automatak/dnp3.git
```

En ocasiones, cuando la distribución linux es recientemente instalada, la herramienta “git” no está instalada. Esta herramienta se encarga de direccionarse al vínculo, descargar el o los archivos, y traerlos respectivamente a la ubicación donde el usuario se encuentra.
Para instalar esta herramienta en caso de ocurrir ese error, solo debe escribirse lo siguiente.

```
root@jose-Latitude-D510:/home/jose# sudo apt-get install git
```

Y posterior a eso se vuelve a repetir la instrucción de clonado para que pueda

descargarse el directorio con la biblioteca en la computadora.

B.2 Configuración y compilación por defecto de los códigos de ejemplo.

Luego de haber descargado el directorio con la biblioteca se debe ingresar a ella desde la línea de comandos con la siguiente instrucción.

```
root@jose-Latitude-D510:/home/jose# cd dnp3
```

Algo muy importante que se debe aclarar, es que la librería trae códigos de ejemplo. De maestros, y estaciones remotas, los cuales pueden ser configurados a beneficio del desarrollador para poder cumplir ciertos fines. En nuestro caso para la interrogación posterior de los medidores, utilizaremos nada más el código “main.cpp de la carpeta master” que correspondería a la configuración del maestro.

Para poder habilitar la función correspondiente a los códigos de ejemplo debemos activar la característica “DEMO” de la biblioteca. Si esto no se realiza, habrá errores en la compilación.

```
> cmake ../dnp3 -D<option>=ON
```

Option Name	Comments
DNP3_ALL	build all optional components below
DNP3_DEMO	example programs
DNP3_JAVA	java bindings shared library
DNP3_TEST	unit test suites
DNP3_TLS	support for TLS channels (requires openssl)
DNP3_DECODER	decoder module

Figura 46. Características disponibles y susceptibles a activar en OpenDNP3 [5].

En la Figura 46 se muestran las características que pueden ser habilitadas, sin embargo para nuestra incumbencia, solo se debe habilitar la característica “DEMO”, que es la correspondiente a los programas de ejemplo.

La característica DEMO, se habilita escribiendo la siguiente línea en la terminal de Linux.

```
root@jose-Latitude-D510:/home/jose# cmake ../dnp3 –DDNP3_DEMO=ON
```

A partir de ese momento puede realizarse ya la primera compilación en la cual se compilaran los programas de ejemplo en los directorios Master, Outstation y Master-GPRS. Para realizar dicha compilación, debemos escribir en la línea de comandos la línea de compilación para CMake.

```
root@jose-Latitude-D510:/home/jose# make
```

Por último se debe completar la instalación con la siguiente línea.

```
root@jose-Latitude-D510:/home/jose# make install
```

El proceso tardara unos minutos pero en este caso también se verá en porcentajes el progreso en la línea de comandos. De esta compilación, surgirán 3 ejecutables correspondientes a los códigos de ejemplo de los directorios ya mencionados. Estos ejecutables hasta el momento no podrán ser utilizados, porque el código “main.cpp” de la carpeta master, está configurado para OpenDNP3 TCP y como ya fue dicho, en el laboratorio, los medidores para DNP3 solo funcionan por medio del puerto Serie.

Debido a que los medidores SHARK están configurados para nada mas funcionar como “esclavos”, o “estaciones remotas”, no será necesario utilizar el código de ejemplo OutStation para comunicarse con él. Sin embargo el código que se modificara y se utilizara, en este caso, tanto en Lubuntu o Raspbian, es el código ejemplo de Master que tiene como nombre main.cpp, Ubicado específicamente en la dirección /dnp3/cpp/examples/master/main.cpp de la biblioteca.

Es importante aclarar que para la compilación con CMake, debe haber un archivo CMakeListis.txt que es el que tiene especificado las rutas donde están las librerías que serán necesarias para la compilación, la ubicación de los ejecutables que se crearan, variables de entorno que sean necesarias, o la inclusión de librerías extra necesarias para diversos proyectos, entre otras cosas. Sin embargo nosotros no realizaremos ninguna modificación en el archivo que la biblioteca trae por defecto, porque así como esta puede utilizarse y por consiguiente funcionar perfectamente.

Anexo C. Código CMakeLists.txt para compilación con CMake.

El código que se muestra a continuación, es el que tiene dentro de sí, la directiva completa de la compilación total de la biblioteca de OpenDNP3. El archivo se denomina CMakeLists.txt y se encuentra en la carpeta principal de de biblioteca /dnp3. El codigo completo se muestra a continuación.

```
1. cmake_minimum_required (VERSION 2.8)
2. project (opendnp3)
3. set(OPENDNP3_MAJOR_VERSION 2)
4. set(OPENDNP3_MINOR_VERSION 2)
5. set(OPENDNP3_MICRO_VERSION 1)
6. set(OPENDNP3_VERSION ${OPENDNP3_MAJOR_VERSION}.${OPENDNP3_MINOR_VER
   SION}.${OPENDNP3_MICRO_VERSION})
7.
8. if(NOT CMAKE_BUILD_TYPE)
9.   set(CMAKE_BUILD_TYPE "Release" CACHE STRING
10.           "Choose the type of build, options are: Debug Release
11.     RelWithDebInfo MinSizeRel."
12.           FORCE)
13.     endif()
14.
15.     message("CMake build is: ${CMAKE_BUILD_TYPE}")
16.     include(${PROJECT_SOURCE_DIR}/cmake/settings.cmake)
17.
18.     # various optional libraries and projects
19.     option(DNP3_ALL "Build all optional projects (secauth, demos,
   tests)" OFF)
20.     option(DNP3_TEST "Build tests" OFF)
21.     option(DNP3_DEMO "Build demo applications" OFF)
22.     option(DNP3_DECODER "Build the decoder library" OFF)
23.     option(DNP3_TLS "Build TLS client/server support" OFF)
24.     option(DNP3_JAVA "Build the Java bindings" OFF)
25.
26.     # other options off-by-default that you can enable
27.     option(WERROR "Set all warnings to errors" OFF)
28.     option(STATICLIBS "Builds static versions of all installed
   libraries" OFF)
29.     option(COVERAGE "Builds the libraries with coverage info for
   gcov (gcc only)" OFF)
30.     option(PROFILE "Builds the libraries with profiling support
   (gcc only)" OFF)
31.
32.     if(DNP3_ALL)
33.             message("enabling all optional components")
34.             set(DNP3_DEMO ON)
35.             set(DNP3_TEST ON)
36.             set(DNP3_TLS ON)
37.             set(DNP3_DECODER ON)
38.             set(DNP3_JAVA ON)
39.     endif()
```

```
40.
41.     if(DNP3_JAVA)
42.             find_package(Java COMPONENTS Development)
43.             find_package(JNI REQUIRED)
44.             include_directories(${JNI_INCLUDE_DIRS})
45.     endif()
46.
47.     SET(ASIO_SUBMODULE_DIR "${PROJECT_SOURCE_DIR}/deps/asio/asio/
   include")
48.     # detection stuff for ASIO
49.     if (EXISTS "${ASIO_SUBMODULE_DIR}/asio.hpp")
50.         message("ASIO has been checked out as a git
   submodule: ${ASIO_SUBMODULE_DIR}")
51.             include_directories(${ASIO_SUBMODULE_DIR})
52.     else()
53.             message("ASIO has NOT been checked out as a git
   submodule...")
54.             if (ASIO_HOME)
55.                     message("ASIO_HOME defined in
   cache: ${ASIO_HOME}")
56.                     include_directories(${ASIO_HOME})
57.             else()
58.                     if(DEFINED ENV{ASIO_HOME})
59.                             message("ASIO_HOME defined in
   environment: $ENV{ASIO_HOME}")
60.                             include_directories($ENV{ASIO_HOME})

61.                     else()
62.                             message("ASIO_HOME was not defined.
   ASIO expected to be on include path")
63.                     endif()
64.             endif()
65.     endif()
66.
67.     # required for ASIO in C++11 only mode
68.     add_definitions(-DASIO_STANDALONE)
69.
70.     if(DNP3_TLS)
71.             add_definitions(-DOPENDNP3_USE_TLS)
72.
73.             find_package(OpenSSL REQUIRED)
74.             message("OpenSSL libraries: ${OPENSSL_LIBRARIES}")
75.             if(WIN32)
76.                     include_directories(${OPENSSL_INCLUDE_DIR})
77.             endif()
78.     endif()
79.
80.     set(CMAKE_REQUIRED_FLAGS ${CMAKE_CXX_FLAGS})
81.     # include paths for all the local libraries
82.     include_directories(./cpp/libs/src)
83.     include_directories(./cpp/libs/include)
84.     include_directories(./cpp/tests/libs/src)
85.     # ---- openpal library ----
```

```
86.     file(GLOB_RECURSE openpal_SRC ./cpp/libs/src/openpal/*.cpp
   ./cpp/libs/src/openpal/*.h ./cpp/libs/include/openpal/*.h)
87.     add_library(openpal ${LIB_TYPE} ${openpal_SRC})
88.     install(TARGETS openpal DESTINATION lib)
89.     set_target_properties(openpal PROPERTIES FOLDER
   libs VERSION ${OPENDNP3_VERSION} SOVERSION ${OPENDNP3_MAJOR_VERSION
   })
90.
91.     # ---- opendnp3 library ----
92.     file(GLOB_RECURSE opendnp3_SRC ./cpp/libs/src/opendnp3/*.cpp
   ./cpp/libs/src/opendnp3/*.h ./cpp/libs/include/opendnp3/*.h)
93.     add_library(opendnp3 ${LIB_TYPE} ${opendnp3_SRC})
94.     target_link_libraries(opendnp3 openpal)
95.     install(TARGETS opendnp3 DESTINATION lib)
96.     set_target_properties(opendnp3 PROPERTIES FOLDER
   libs VERSION ${OPENDNP3_VERSION} SOVERSION ${OPENDNP3_MAJOR_VERSION
   })
97.
98.     if(DNP3_DECODER)
99.       file(GLOB_RECURSE dnp3decode_SRC
   ./cpp/libs/src/dnp3decode/*.cpp ./cpp/libs/src/dnp3decode/*.h
   ./cpp/libs/include/dnp3decode/*.h)
100.      add_library(dnp3decode ${LIB_TYPE} ${dnp3decode_SRC})
101.      target_link_libraries(dnp3decode opendnp3)
102.      install(TARGETS dnp3decode DESTINATION lib)
103.      set_target_properties(dnp3decode PROPERTIES FOLDER
   libs VERSION ${OPENDNP3_VERSION} SOVERSION ${OPENDNP3_MAJOR_VERSION
   })
104.    endif()
105.
106.    # ---- asiopal library ----
107.    file(GLOB_RECURSE asiopal_HPP ./cpp/libs/src/asiopal/*.h
   ./cpp/libs/include/asiopal/*.h)
108.
109.    if(DNP3_TLS)
110.      file(GLOB_RECURSE asiopal_CPP
   ./cpp/libs/src/asiopal/*.cpp)
111.    else()
112.      file(GLOB asiopal_CPP ./cpp/libs/src/asiopal/*.cpp)
113.    endif()
114.
115.    add_library(asiopal ${LIB_TYPE} ${asiopal_HPP} ${asiopal_CPP}
   )
116.
117.    set(asiopal_link_libraries "openpal")
118.    if(DNP3_TLS)
119.        set(asiopal_link_libraries "${asiopal_link_libraries};${O
   PENSSL_LIBRARIES}")
120.    endif()
121.
122.    target_link_libraries(asiopal ${asiopal_link_libraries})
123.    install(TARGETS asiopal DESTINATION lib)
```

```
124.     set_target_properties(asiopal PROPERTIES FOLDER
   libs VERSION ${OPENDNP3_VERSION} SOVERSION ${OPENDNP3_MAJOR_VERSION
   })
125.
126.     # ---- asiodnp3 library ----
127.     file(GLOB_RECURSE asiodnp3_HPP ./cpp/libs/src/asiodnp3/*.h
   ./cpp/libs/include/asiodnp3/*.h)
128.     if(DNP3_TLS)
129.       file(GLOB_RECURSE asiodnp3_CPP
   ./cpp/libs/src/asiodnp3/*.cpp)
130.     else()
131.       file(GLOB asiodnp3_CPP ./cpp/libs/src/asiodnp3/*.cpp)
132.     endif()
133.     add_library(asiodnp3 ${LIB_TYPE} ${asiodnp3_HPP} ${asiodnp3_C
   PP})
134.     target_link_libraries(asiodnp3 asiopal opendnp3)
135.     install(TARGETS asiodnp3 DESTINATION lib)
136.     set_target_properties(asiodnp3 PROPERTIES FOLDER
   libs VERSION ${OPENDNP3_VERSION} SOVERSION ${OPENDNP3_MAJOR_VERSION
   })
137.
138.     if(DNP3_JAVA)
139.       file(GLOB_RECURSE opendnp3java_SRC ./java/cpp/*.h
   ./java/cpp/*.cpp)
140.       add_library(opendnp3java SHARED ${opendnp3java_SRC})
141.       target_link_libraries(opendnp3java asiodnp3)
142.       install(TARGETS opendnp3java DESTINATION lib)
143.       set_target_properties(opendnp3java PROPERTIES FOLDER
   libs VERSION ${OPENDNP3_VERSION} SOVERSION ${OPENDNP3_MAJOR_VERSION
   })
144.     endif()
145.
146.     # ----- install -----
147.     # common pattern and exludes for all installed headers
148.     set(INSTALL_ARGS FILES_MATCHING
   PATTERN "*.h" PATTERN ".deps" EXCLUDE PATTERN ".libs" EXCLUDE)
149.     install(DIRECTORY ./cpp/libs/include/ DESTINATION
   include ${INSTALL_ARGS})
150.
151.     if(DNP3_DEMO)
152.
153.       # ----- master demo executable -----
154.       add_executable(master-demo ./cpp/examples/master/main.cpp)
155.       target_link_libraries (master-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
156.       set_target_properties(master-demo PROPERTIES FOLDER demos)
157.
158.       # ----- master demo executable -----
159.       add_executable(master-gprs-demo ./cpp/examples/master-
   gprs/main.cpp)
160.       target_link_libraries (master-gprs-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
```

```
161.       set_target_properties(master-gprs-demo PROPERTIES FOLDER
   demos)
162.
163.       # ----- outstation demo executable -----
164.       add_executable(outstation-demo
   ./cpp/examples/outstation/main.cpp)
165.       target_link_libraries (outstation-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
166.       set_target_properties(outstation-demo PROPERTIES FOLDER
   demos)
167.
168.       if(DNP3_DECODER)
169.
170.         # ----- decoder executable -----
171.         add_executable(decoder ./cpp/examples/decoder/main.cpp)
172.         target_link_libraries (decoder asiodnp3
   dnp3decode ${PTHREAD})
173.         set_target_properties(decoder PROPERTIES FOLDER demos)
174.
175.       endif()
176.
177.       if(DNP3_TLS)
178.             # ----- master tls executable -----
179.             add_executable(master-tls-demo
   ./cpp/examples/tls/master/main.cpp)
180.             target_link_libraries (master-tls-demo LINK_PUBLIC
   asiodnp3 ${PTHREAD})
181.             set_target_properties(master-tls-demo PROPERTIES
   FOLDER demos/tls)
182.
183.             # ----- outstation tls executable -----
184.             add_executable(outstation-tls-demo
   ./cpp/examples/tls/outstation/main.cpp)
185.             target_link_libraries (outstation-tls-demo
   LINK_PUBLIC asiodnp3 ${PTHREAD})
186.             set_target_properties(outstation-tls-demo PROPERTIES
   FOLDER demos/tls)
187.
188.             # ----- master-gprs tls executable -----
189.             add_executable(master-gprs-tls-demo
   ./cpp/examples/tls/master-gprs/main.cpp)
190.             target_link_libraries (master-gprs-tls-demo
   LINK_PUBLIC asiodnp3 ${PTHREAD})
191.             set_target_properties(master-gprs-tls-demo PROPERTIES
   FOLDER demos/tls)
192.
193.        endif()
194.
195.     endif()
196.     if(DNP3_TEST)
197.
198.       enable_testing()
199.       # ----- testlib library ------
```

```
200.       file(GLOB_RECURSE testlib_SRC
   ./cpp/tests/libs/src/testlib/*.cpp
   ./cpp/tests/libs/src/testlib/*.h)
201.       add_library(testlib ${testlib_SRC})
202.       target_link_libraries(testlib openpal)
203.       set_target_properties(testlib PROPERTIES FOLDER
   tests/mocks)
204.
205.       # ----- dnp3mocks library ------
206.       file(GLOB_RECURSE dnp3mocks_SRC
   ./cpp/tests/libs/src/dnp3mocks/*.cpp
   ./cpp/tests/libs/src/dnp3mocks/*.h)
207.       add_library(dnp3mocks ${dnp3mocks_SRC})
208.       target_link_libraries(dnp3mocks opendnp3 testlib)
209.       set_target_properties(dnp3mocks PROPERTIES FOLDER
   tests/mocks)
210.
211.       # ----- openpal tests -----
212.       file(GLOB_RECURSE openpal_TESTSRC
   ./cpp/tests/openpal/src/*.cpp ./cpp/tests/openpal/src/*.h)
213.       add_executable (testopenpal ${openpal_TESTSRC})
214.       target_link_libraries (testopenpal LINK_PUBLIC
   testlib ${PTHREAD})
215.       set_target_properties(testopenpal PROPERTIES FOLDER tests)
216.       add_test(testopenpal testopenpal)
217.
218.       # ----- opendnp3 tests -----
219.       file(GLOB_RECURSE opendnp3_TESTSRC
   ./cpp/tests/opendnp3/src/*.cpp ./cpp/tests/opendnp3/src/*.h)
220.       add_executable (testopendnp3 ${opendnp3_TESTSRC})
221.       target_link_libraries (testopendnp3 LINK_PUBLIC
   dnp3mocks ${PTHREAD})
222.       set_target_properties(testopendnp3 PROPERTIES FOLDER tests)
223.       add_test(testopendnp3 testopendnp3)
224.
225.       # ----- asiopal tests -----
226.       if(DNP3_TLS)
227.         file(GLOB_RECURSE asiopal_TESTSRC
   ./cpp/tests/asiopal/src/*.cpp ./cpp/tests/asiopal/src/*.h)
228.       else()
229.         file(GLOB asiopal_TESTSRC ./cpp/tests/asiopal/src/*.cpp
   ./cpp/tests/asiopal/src/*.h ./cpp/tests/asiopal/src/mocks/*.cpp
   ./cpp/tests/asiopal/src/mocks/*.h)
230.       endif()
231.       add_executable (testasiopal ${asiopal_TESTSRC})
232.       target_link_libraries (testasiopal LINK_PUBLIC asiopal
   testlib ${PTHREAD})
233.       set_target_properties(testasiopal PROPERTIES FOLDER tests)
234.       add_test(testasiopal testasiopal)
235.
236.       # ----- asiodnp3 tests -----
237.       file(GLOB_RECURSE asiodnp3_TESTSRC
   ./cpp/tests/asiodnp3/src/*.cpp ./cpp/tests/asiodnp3/src/*.h)
```

```
238.       add_executable (testasiodnp3 ${asiodnp3_TESTSRC})
239.       target_link_libraries (testasiodnp3 LINK_PUBLIC asiodnp3
   dnp3mocks ${PTHREAD})
240.       set_target_properties(testasiodnp3 PROPERTIES FOLDER tests)
241.       add_test(testasiodnp3 testasiodnp3)
242.
243.       # ----- fuzz tests -----
244.       if(DNP3_DECODER)
245.         add_executable (fuzzdnp3 ./cpp/tests/fuzz/onefile.cpp
   ./cpp/tests/fuzz/fuzzdnp3.cpp)
246.         target_link_libraries (fuzzdnp3 asiodnp3
   dnp3decode ${PTHREAD})
247.         set_target_properties(fuzzdnp3 PROPERTIES FOLDER tests)
248.       endif()
249.
250.     endif()
251.
252.     add_custom_target(
253.             format
254.             WORKING_DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR}
255.             COMMAND astyle -R ./cpp/*.h ./cpp/*.cpp --
   options=./config/astyle.cfg --
   exclude=./cpp/libs/include/opendnp3/gen --
   exclude=./cpp/libs/src/opendnp3/gen --
   exclude=./cpp/libs/src/opendnp3/objects
256.             COMMAND astyle -R ./java/cpp/adapters/*.h
   ./java/cpp/adapters/*.cpp --options=./config/astyle.cfg
257.     )
```

Figura 47. Código CMakeLists.txt.

Anexo D. Mapa DNP de medidores SHARK 200 s completo.

Object	Point	Var	Description	Format	Range	Multiplier	Units	Comments
80	7	1	Device Restart Bit	N/A	N/A	N/A	none	Clear via Function 2 (Write), Qualifier Code 0.

Figura 48. Grupo de objetos 80 destinado para el reinicio programado del medidor [9].

Object	Point	Var	Description	Format	Range	Multiplier	Units	Comments
10	0	2	Reset Energy Counters	BYTE	Always 1	N/A	None	Read by Class 0 or with qualifier 0, 1, 2, or 6
10	1	2	Change to Modbus RTU Protocol	BYTE	Always 1	N/A	None	Read by Class 0 or with qualifier 0, 1, 2, or 6
10	2	2	Reset Demand Cntrs (Max / Min)	BYTE	Always 1	N/A	None	Read by Class 0 or with qualifier 0, 1, 2, or 6

Figura 49. Grupo de objetos correspondientes a las salidas binarias [9]

Object	Point	Var	Description	Format	Range	Multiplier	Units	Comments
12	0	1	Reset Energy Counters	N/A	N/A	N/A	none	Responds to Function 5 (Direct Operate), Quali- fier Code 17x or 28x, Control Code 3, Count 0, On 0 msec, Off 1 msec ONLY.
12	1	1	Change to Modbus RTU Protocol	N/A	N/A	N/A	none	Responds to Function 6 (Direct Operate - No Ack), Qualifier Code 17x, Control Code 3, Count 0, On 0 msec, Off 1 msec ONLY.
12	2	1	Reset Demand Counters (Max / Min)	N/A	N/A	N/A	none	Responds to Function 5 (Direct Operate), Quali- fier Code 17x or 28x, Control Code 3, Count 0, On 0 msec, Off 1 msec ONLY.

Figura 50. Grupo de objetos correspondientes a las salidas relevadoras de control [9].

Object	Point	Var	Description	Format	Range	Multiplier	Units	Comments
20	0	5	W-hours, Positive	UINT32	0 to 99999999	Multiplier = 10(n-d), where n and d are derived from the energyformat. n = 0, 3, or 6 per energy format scale and d = number of decimal places.	Whr	example: energy format = 7.2K and W- hours counter = 1234567 n=3 (K scale), d=2 (2 digits after deci- mal point), mul-tiplier = 10(3-2) = 101 = 10, so energy is 1234567 * 10 Whrs, or 12345.67 KWhrs
20	1	5	W-hours, Negative	UINT32	0 to 99999999		Whr	
20	2	5	VAR-hours, Positive	UINT32	0 to 99999999		VARhr	
20	3	5	VAR-hours, Negative	UINT32	0 to 99999999		VARhr	
20	4	5	VA-hours, Total	UINT32	0 to 99999999		VAhr	

Figura 51. Grupo de objetos correspondientes a contadores binarios [9].

Object	Point	Var	Description	Format	Range	Multiplier	Units	Comments
30	0	4	Meter Health	sint16	0 or 1	N/A	None	0 = OK
30	1	4	Volts A-N	sint16	0 to 32767	(150 / 32768)	V	Values above 150V secondary read 32767.
30	2	4	Volts B-N	sint16	0 to 32767	(150 / 32768)	V	
30	3	4	Volts C-N	sint16	0 to 32767	(150 / 32768)	V	

30	4	4	Volts A-B	sint16	0 to 32767	(300 / 32768)	V	Values above 300V secondary read 32767.
30	5	4	Volts B-C	sint16	0 to 32767	(300 / 32768)	V	
30	6	4	Volts C-A	sint16	0 to 32767	(300 / 32768)	V	
30	7	4	Amps A	sint16	0 to 32767	(10 / 32768)	A	Values above 10A secondary read 32767.
30	8	4	Amps B	sint16	0 to 32767	(10 / 32768)	A	
30	9	4	Amps C	sint16	0 to 32767	(10 / 32768)	A	
30	10	4	Watts, 3-Ph total	sint16	-32768 to +32767	(4500 / 32768)	W	
30	11	4	VARs, 3-Ph total	sint16	-32768 to +32767	(4500 / 32768)	VAR	
30	12	4	VAs, 3-Ph total	sint16	0 to +32767	(4500 / 32768)	VA	
30	13	4	Power Factor, 3-Ph total	sint16	-1000 to +1000	0.001	None	
30	14	4	Frequency	sint16	0 to 9999	0.01	Hz	
30	15	4	Positive Watts, 3-Ph, Maximum Avg Demand	sint16	-32768 to +32767	(4500 / 32768)	W	
30	16	4	Positive VARs, 3-Ph, Maximum Avg Demand	sint16	-32768 to +32767	(4500 / 32768)	VAR	
30	17	4	Negative Watts, 3-Ph, Maximum Avg Demand	sint16	-32768 to +32767	(4500 / 32768)	W	
30	18	4	Negative VARs, 3-Ph, Maximum Avg Demand	sint16	-32768 to +32767	(4500 / 32768)	VAR	
30	19	4	VAs, 3-Ph, Maximum Avg Demand	sint16	-32768 to +32767	(4500 / 32768)	VA	
30	20	4	Angle, Phase A Current	sint16	-1800 to +1800	0.1	degree	
30	21	4	Angle, Phase B Current	sint16	-1800 to +1800	0.1	degree	

30	22	4	Angle, Phase C Current	sint16	-1800 to +1800	0.1	degree	
30	23	4	Angle, Volts A-B	sint16	-1800 to +1800	0.1	degree	
30	24	4	Angle, Volts B-C	sint16	-1800 to +1800	0.1	degree	
30	25	4	Angle, Volts C-A	sint16	-1800 to +1800	0.1	degree	
30	26	4	CT numerator	sint16	1 to 9999	N/A	none	CT ratio = (numerator * multiplier) / denomina-tor
30	27	4	CT multiplier	sint16	1, 10, or 100	N/A	none	
30	28	4	CT denominator	sint16	1 or 5	N/A	none	
30	29	4	PT numerator	SINT16	1 to 9999	N/A	none	PT ratio = (numerator * multiplier) / denomina-tor
30	30	4	PT multiplier	SINT16	1, 10, or 100	N/A	none	
30	31	4	PT denominator	SINT16	1 to 9999	N/A	none	
30	32	4	Neutral Current	SINT16	0 to 32767	(10 / 32768)	A	For 1A model, mul- tiplier is (2 / 32768) and values above 2A secondary read 32767

Figura 52. Grupo de objetos de entradas analógicas [9].

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.* Trabajo de graduación, Universidad de El Salvador. [Online]
http://ri.ues.edu.sv/5584/

[2] Pagina oficial de Automatak OpenDNP3. [Online]
https://www.automatak.com/opendnp3/

[3] Guía OpenDNP3. Página principal.
https://www.automatak.com/opendnp3/docs/guide/current/

[4] Documentación de referencia para biblioteca OpenDNP3. [Online]
https://www.automatak.com/opendnp3/docs/cpp/current/

[5] Guía OpenDNP3. En apartado Build. CMake. [Online]
https://www.automatak.com/opendnp3/docs/guide/current/build/cmake/

[6] Sitio de descarga de software de simulación de OpenDNP3 [Online]
Https://automatak.com/opendnp3/simulator/setup.msi

[7] Funcion Modbus _read_registers(). Libmodbus documentación [Online].
http://libmodbus.org/docs/v3.0.6/modbus_read_registers.html

[8] Funcion Modbus _new_RTU(). Libmodbus documentación [Online].
_http://libmodbus.org/docs/v3.0.6/modbus_new_rtu.html

[9] *Electro industries* SHARK 200S. Manual De Instalación y Operación. [Online]
https://www.electroind.com/pdf/sp/01_29_13/ES149721_Shark_200S_man_sp.pdf

[10] Sitio de descarga de *Comunicator Ext*. *Electro industries*. [Online]
https://marketing.electroind.com/acton/form/25598/0028:d-0001/0/-/-/-/-/index.htm

[11] Duque Alas, Alexander Omar (2014) *Optimización del sistema de monitorización remota de medidores de energía eléctrica.* Trabajo de graduación, Universidad de El Salvador. [Online]
http://ri.ues.edu.sv/6534/

[12] DNP3. Capa de enlace.
https://es.wikipedia.org/wiki/DNP3

[13] DNP3. Protocolo DNP3.
https://es.scribd.com/document/192535757/protocolo-DNP3

[14] Renderos Alfaro, Samuel Antonio. Estudio e implementación del protocolo DNP3 sobre sistemas embebidos. Proyecto de Ingeniería Eléctrica, Universidad de El Salvador. [Online]
http://www.academia.edu/22257073/UNIVERSIDAD_DE_EL_SALVADOR_FACULTAD_DE_INGENIERIA_Y_ARQUITECTURA_ESCUELA_DE_INGENIERIA_ELECTRICA_PROYECTO_DE_INGENIERIA_I_ESTUDIO_E_IMPLEMENTACION_DEL_PROTOCOLO_DNP3_SOBRE_SISTEMAS_EMBEBIDOS_2015

[15] Palacios Girón, Luis Humberto. *Ampliación del monitoreo de variables eléctricas en las subestaciones de la Universidad de El Salvador.* Trabajo de graduación, Universidad de El Salvador. [Online]
http://ri.ues.edu.sv/15091/