UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Aplicación del router Dragino como dispositivo de almacenamiento masivo.

PRESENTADO POR:

MARVIN ANDRÉS CORNEJO MEJÍA

PARA OPTAR AL TITULO DE:

INGENIERO ELECTRICISTA

CIUDAD UNIVERSITARIA, MARZO DE 2014

UNIVERSIDAD DE EL SALVADOR

RECTOR :

ING. MARIO ROBERTO NIETO LOVO

SECRETARIA GENERAL :

DRA. ANA LETICIA ZAVALETA DE AMAYA

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO :

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO :

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA ELÉCTRICA

DIRECTOR :

ING. JOSÉ WILBER CALDERÓN URRUTIA

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE INGENIERÍA ELÉCTRICA

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO ELECTRICISTA

Título :

Aplicación del router Dragino como dispositivo de almacenamiento masivo.

Presentado por :

MARVIN ANDRÉS CORNEJO MEJÍA

Trabajo de Graduación Aprobado por:

Docente Director :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

San Salvador, Marzo 2014

Trabajo de Graduación Aprobado por:

Docente Director :

Dr. CARLOS EUGENIO MARTÍNEZ CRUZ

ACTA DE CONSTANCIA DE NOTA Y DEFENSA FINAL

En esta fecha, 6 de marzo de 2014, en la Sala de Lectura de la Escuela de Ingeniería Eléctrica, a las 3:00 horas, en presencia de las siguientes autoridades de la Escuela de Ingeniería Eléctrica de la Universidad de El Salvador:

1. Ing. José Wilber Calderón Urrutia
   Director — Firma: _____

2. Ing. Salvador de Jesús Germán
   Secretario — Firma: _____

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

1- Ing. Walter Leopoldo Zelaya Chicas — Firma: _____

2- Ing. Omar Otoniel Flores Cortez — _____

Se efectuó la defensa final reglamentaria del Trabajo de Graduación:

Aplicación del router Dragino como dispositivo de almacenamiento masivo.

A cargo del Bachiller:

- Cornejo Mejía, Marvín Andrés

Habiendo obtenido en el presente Trabajo una nota promedio de la defensa final, de: 8.5

(OCHO PUNTO CINCO)

Agradecimientos.

A Dios, nuestro Señor, por haberme dado vida, por sus bendiciones para culminar una etapa más de mi vida, a nuestra madre María Santísima por estar siempre conmigo en todo momento.

A mis tíos, Mario, Delmy y Reynaldo, por su apoyo y ánimos durante toda mi vida, especialmente en la etapa de mi carrera universitaria. Gracias por confiar en mí, les estaré eternamente agradecido por todo.

A Sussana Castellanos, por estar a mi lado durante todo este proceso, por darme ánimos de seguir adelante y nunca abandonarme en este largo camino. Gracias por todo tu apoyo.

A mi asesor por ser parte de este esfuerzo, gracias por el apoyo brindado a lo largo del proyecto de graduación.

A todos que de alguna forma fueron parte de todo este proceso y me ayudaron a concluir esta etapa de mi vida. Gracias.

Marvin Andrés Cornejo Mejía.

En memoria de mis abuelitos Ángela y Pedro, mi padre Carlos.
Que en paz descansen.

OBJETIVOS

Objetivo general:

Construir un prototipo que integre el router Dragino con una placa Arduino para el almacenamiento masivo de datos.

Objetivos específicos:

- Realizar almacenamiento de datos mediante una placa Arduino y un módulo de memoria SD.
- Intercomunicar el router Dragino con la placa Arduino.
- Dotar al router Dragino de capacidad de almacenamiento masivo.
- Utilizar técnicas de compilación cruzada en el desarrollo de programación dentro del router Dragino.
- Desarrollar aplicaciones en el router Dragino basadas en la librería LibModbus.
- Sustituir aplicaciones basadas en computadora personal por dispositivos empotrados.
- Dotar a las redes de monitoreo de la capacidad de respaldo.
- Reducir el consumo de energía eléctrica mediante la incorporación de dispositivos de bajo consumo de energía eléctrica.

CONTENIDO

1.0. INTRODUCCION. 18
1.1 DESCRIPCION DEL PROYECTO. 19
1.2. COMPILACIÓN CRUZADA. 20
1.3. LENGUAJES DE PROGRAMACION UTILIZADOS. 21
2.0. DISEÑO HARDWARE DEL PROTOTIPO. 23
2.1. MEDIDOR SHARK Y ROUTER DRAGINO. 23
2.2. COMUNICACION ENTRE ROUTER DRAGINO Y ARDUINO. 24
2.3. CONFIGURACION DE PUERTO UART DEL ROUTER DRAGINO. 25
2.4. PLACA ARDUINO CON DATALOGGER SD. 27
3.0. CONFIGURACIÓN SOFTWARE. 29
3.1. CONFIGURACION DE PC PARA BUILDROOT. 29
3.3. COMPILACION CRUZADA CON BUILDROOT. 39
3.4. LIBMODBUS EN ROUTER DRAGINO. 40
3.4. CODIGO LUA. 42
3.5. CODIGO ARDUINO. 43
4.0. COMUNICACION ENTRE DISPOSITIVO DE ALMACENAMIENTO Y RED DE MEDIDORES DE LA UES. 48
4.1. INSTALACION DE BATMAND. 49
4.2. CONFIGURACIÓN DE ROUTER DRAGINO COMO SUPER NODO. 51
4.3. PRUEBAS DE COMUNICACIÓN ENTRE ROUTER DRAGINO Y RED DE MEDIDORES. 55
4.5. AUMENTO DE CAPACIDAD DE BUFFER EN PUERTO SERIAL DE ARDUINO. 58
CONCLUSIONES 61
ANEXO A. 62
A.1. ROUTER DRAGINO. 62
A.2. INSTALACION DE FIRMWARE ROUTER DRAGINO. 65
A.3. PASOS PREVIOS A INSTALACION DE FIRMWARE. 66
A.4. INSTALACIONDE FIRMWARE DRAGINO. 68
A.5. ACCESO A ROUTER DRAGINO. 69
A.6. PLACA ARDUINO. 70

A.7. DATALOGGER SD. 70
A.8. INSTALACION DE ARDUINO. 72
A.9. EJECUCIÓN DE PROGRAMA ARDUINO. 72
ANEXO B. 74
B.1. LIBRERIA LIBMODBUS. 74
B.2. FUNCIONES EN LIBMODBUS. 74
B.3. PROTOCOLO MODBUS. 75
B.4. INSTALACION DE LIBMODBUS EN PC. 75
B.5. COMUNICACIÓN Y DISPOSITIVOS 77
B.6. MODELO CLIENTE – SERVIDOR (MAESTRO/ESCLAVO) 77
B.7. ESTABLECIMIENTO DE LA CONEXIÓN. 78
B.8. TRANSFERENCIA DE DATOS DE MODBUS 79
B.9. UTILIZACION DE WIRESHARK PARA TRAFICO DE DATOS. 80
B.9. INSTALACIÓN DE LUA Y LUA SOCKET EN PC. 83
B.10. INSTALACIÓN DE LUA SOCKET EN ROUTER DRAGINO. 84
ANEXO C. 85
C.1. MEDIDOR SHARK. 85
C.2. FUNCIONES BASICAS. 85
C.3. CARACTERISTICAS TECNICAS. 86
C.4. MAPA MODBUS EN SHARK. 87
C.5. FORMATO DE DATOS. 88
C.6. MAPA MODBUS. 89
ANEXO D. 93
D.1 INSTALACION DE SIMULADOR DIAGSLAVE. 93
D.2. PRUEBAS REALIZADAS CON EL SIMULADOR DIAGSLAVE Y ROUTER DRAGINO. 94
D.3. ANALISIS DE RESULTADOS CON WIRESHARK. 96
BIBLIOGRAFIA. 100

LISTA DE FIGURAS:

Figura 1.1. Diagrama de flujo del proyecto.....18

Figura 1. 2. Ejecución de Helloword en Router Dragino…….....20

Figura 1. 3. Error al ejecutar código con librerias Modbus……...21

Figura 2.1. Esquema del prototipo de almacenamiento masivo de datos.....23

Figura 2.2. Medidores Shark con router Dragino.....24

Figura 2.3. Hardware utilizado en el proyecto.....25

Figura 2.4. Ubicación de ttyS0.....26

Figura 2.5. Permisos de ejecución para ttyS0.....26

Figura 2.6. Opciones de ttyS0……26

Figura 2.7. Cambio de baud rate para ttyS.....27

Figura 3.1. Comando para ingresar a menu de configuracion de Buildroot……...31

Figura 3.2. Menú de configuración de Buildroot..... 31

Figura 3.3. Configuración de Libmodbus en Buildroot.....32

Figura 3.4. Guardando configuración de Buildroot ……..…...33

Figura 3.5. Línea para poder realizar la configuración de Buildroot.....33

Figura 3.6. Exportación de rutas para gcc.....34

Figura 3.7. Descomprimir OPenWrt SDK……....34

Figura 3.8. Carpetas de OpenWrt SDK atheros 2.6.....35

Figura 3.9. Esquema del orden de los Makefile.....35

Figura 3.10. Configuración de Makefile(1).....36

Figura 3.11. Makefile para archivo C.....37

Figura 3.12. Makefile para archivo C++.....37

Figura 3.13. Ejecución de Makefile para compilación cruzada.....38

Figura 3.14. a) Carpeta helloworld del código C/C++. b) archivos generados al compilar código heloworld.c/cpp…….....…38

Figura 3.15. Compilación cruzada con Buildroot.....39

Figura 3.16. Parte del script libmodbus.la.....40

Figura 3.17.Ubicación de libmodbus.so.5 en router Dragino…….....41

Figura 3.18. Script para ejecución de protocolo Modbus en Dragino……...41

Figura 3.19. Código write_uart.lua…….....42

Figura 3.20. Conexión Dragino-Arduino……..42

Figura 3.21. Fragmento de código Arduino para almacenar datos en SD……..42

Figura 3.22. Función loop para recepción de datos desde UART Dragino…....46

Figura 4.1. Topología de conexión de ruter Dragino y red de medidores…….....48

Figura 4.2. Ejemplo de cómo copiar los archivos al router Dragino.....49

Figura 4.3. Instalación de archivos en router Dragino.....50

Figura 4.4. batmand y vis habilitados.....50

Figura 4.5. Borrado de IP TABLES.....50

Figura 4.6. Reinicio del sistema.....51

Figura 4.7. Archivo de configuración wireless.....51

Figura 4.8. Archivo de configuración network……..…..52

Figura 4.9. Archivo de configuración batmand. …….....53

Figura 4.10. Vista de batmand y vis en los procesos del Router Dragino.....53

Figura 4.11. Ejecución de batmand –cd1.....54

Figura 4.12. Ping a 192.168.1.100……...55

Figura 4.13. Ping a medidor de Agronomía…….....55
Figura 4.14. Ping a medidor de Agronomía por interfaz “alias”……..…..56
Figura 4.15. Compilación cruzada para medicina.c…….....56
Figura 4.16. Copia desde pc a router Dragino…….....56
Figura 4.17. Uso de crontab para ejecución de script write_uart.lua…….....57
Figura 4.18. Ejecución de script Lua cada cinco minutos…….....57
Figura 4.19. Ejecución de ssh para ingresar a router Dragino…….....57
Figura 4.20. Datos almacenados en archivo Logger04.csv…….....58
Figura 4.21. Modificación de archivo boards.txt en Arduino.....59
Figura 4.22. Modificación de board en Arduino……..……..…...60
Figura A.1. Router Dragino.....62
Figura A.2. Arquitectura interna del Dragino MS12…….……...63
Figura A.3. Dirección de descarga de ap-51-flash……....65
Figura A.4. Descarga de archivos para instalación…….....65
Figura A.5. Firmware Dragino.....66
Figura A.6. Directorio con archivos de instalación.....66
Figura A.7. Permisos de ejecución de ap51-flash.....67
Figura A.8. Conexión de Dragino con PC……...67
Figura A.9. Configuración de IP en PC. ……..….67
Figura A.10. Instalación de Dragino…….…68
Figura A.11. Detección de router en el proceso de instalación……..…..68
Figura A.12. Proceso de instalación de software……..68
Figura A.13. Interfaz gráfica de Dragino.....69
Figura A.14. Vista de acceso mediante ssh.....69

Figura A.15. Placa Arduino UNO……..…70
Figura A.16. Datalogger SD…….…….....71
Figura A.17. Instalación de Aduino desde línea de comando…….....72
Figura A.18. Interfaz gráfica de Arduino IDE……..……...…72
Figura A.19. Abrir archivo Arduino……..…….....73
Figura A.20. Selección de puerto serie…….…….....73
Figura B.1. Configuración de Libmodbus…….....75
Figura B.2. Ejecución de make para Libmodbus……..…….....76
Figura B.3. Instalación de Libmodbus.....76
Figura B.4. Modbus cliente-servidor…….…….....78
Figura B.5. Conexión de puertos……..…….....78
Figura B.6. Conexión TCP…….……..…….…….....80
Figura B.7. Protocolo de transmisión SYN en Wirwshark….....81
Figura B.8. Protocolo de transmisión SYN, ASK en Wirwshark…..…….…..82
Figura B.9. Protocolo de transmisión SYN en Wirwshark……..…….....82
Figura B.10. Instalación de lua desde terminal.....83
Figura B.11. Instalación de lua5.1.tar.gz……..…….....83
Figura B.12. Instalación de lua socket……..……...…84
Figura B.13. Envío de paquete de PC a router con scp…..…….....84
Figura B.14. Instalación de luasocket en router Dragino……..…84
Figura C.1. Medidor shark 200……...…85
Figura D.1. Ejecución de diagslave en Linux.…….…….....91
Figura D.2. Diagslave desde una Terminal en Linux..……...…94
Figura D.3. Compilación cruzada utilizando herramienta Buildroot...…….…...95

Figura D.4. Comando scp para copiar a router Dragino……..…….....95

Figura D.5. Configuración de IP en la computadora.....95

Figura D.6. Ejecución de ssh para acceso a router Dragino.....95

Figura D.7. Ejecución de sensor011 en router Dragino.....96

Figura D.8. Selección de red a analizar por medio de wireshark.....97

Figura D.9. Trafico de paquetes entre Dragino y pc……..…….....97

Figura D.10. Petición de sincronización de Dragino con pc..…….....98

Figura D.11. Cierre de conexión.…….....98

Figura D.12. Sub menu Flow Graph…….....99

Figura D.13. Grafico del tráfico de paquetes entre simulador Diagslave y router Dragino…….…….....99

Lista de Tablas:

Tabla 2.1. Conexión de pines entre Arduino y Dragino. 24

Tabla 2.2 . Conexión SPI..... 28

Tabla 3.1. Paquetes para instalación de Buildroot en pc. 30

Tabla 3.2. Funciones utilizadas por SD.h..... 44

Tabla A.1. Especificaciones técnicas de router dragino. 63

Tabla C.1. Información sobre el medidor. 89

Tabla C.2. Sección de datos del medidor..... 90

Tabla C.3. Sección de datos del medidor..... 91

ACRONIMOS.

OpenWrt: OpenWrt es una distribución de Linux basada en firmware usada para dispositivos empotrados tales como routers inalámbricos.

WIFI: es una marca de la Wi-Fi Alliance, la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local.

UART: son las siglas de "Universal Asynchronous Receiver-Transmitter" (en español, Transmisor-Receptor Asíncrono Universal). Éste controla los puertos y dispositivos serie.

BAUDIOS: El baudio es una unidad de medida, usada en telecomunicaciones, que representa el número de símbolos transmitidos por segundo en una red analógica.

SIN: pin utilizado para la recepción de señal en router Dragino.

SOUT: pin utilizado para la transmisión de señal en router Dragino.

RX: pin de recepción de señal en placa Arduino.

TX: pin de transmisión de señal en placa Arduino.

TCP: protocolo de control de transmisión por sus siglas en inglés: Transmission Control Protocol.

RTU: unidad terminal remota por sus siglas en inglés: Remote Terminal Unit.

IP: es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz de un dispositivo dentro de una red.

AP: punto de acceso por sus siglas en inglés: Wireless Access Point.

UES: Universidad de El Salvador

1.0. INTRODUCCION.

El fin del presente trabajo es construir un prototipo que integre un router Dragino[1] y una placa Arduino[2] para el almacenamiento masivo de datos. El prototipo se utilizará como dispositivo de almacenamiento de los datos que se registran a diario en la red de medidores de energía, colocados en diversos puntos en el campus de la Universidad de El Salvador.

El trabajo está comprendido de tres partes. La primera parte consiste en la realización de compilación cruzada [3], utilizando la herramienta Buildroot[4]. La segunda parte consiste en la configuración del router Dragino, para tener la capacidad de poder compilar archivos en lenguaje C. Esto debido a que se utilizó la biblioteca de funciones Libmodbus[5], que está configurado en dicho lenguaje. La tercera parte consistió en la programación del módulo Arduino para poder comunicarse con un Datalogger que a su vez almacenara los datos obtenidos del protocolo Modbus.

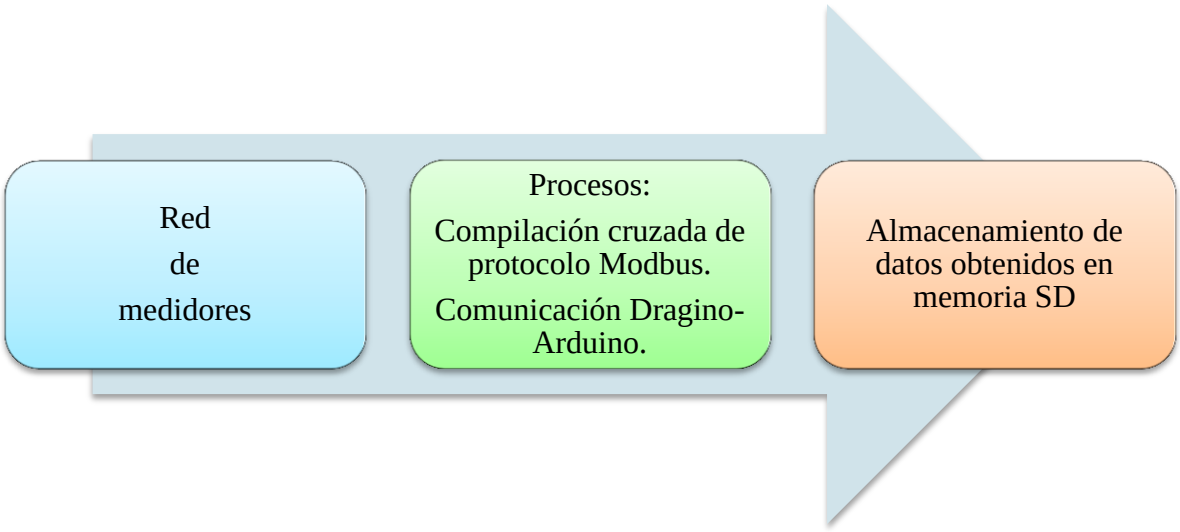


Figura2.1. Diagrama de flujo del proyecto.

1.1 DESCRIPCION DEL PROYECTO.

Se inicia con la toma de datos de la red de medidores que están instalados en el campus de la universidad. La adquisición de datos se realiza a través del protocolo Modbus. Este se utiliza para interrogar a cada medidor donde se obtienen los diferentes parámetros de consumo de energía.

Se utiliza el router Dragino ya que este se puede considerar como una pequeña computadora, por su capacidad de procesamiento, comunicación wifi, puerto Ethernet y puerto UART[6]. Lo cual lo hacen adecuado para fines de investigación. En el apéndice A se dan más detalles sobre este dispositivo.

Los programas de comunicación Modbus se ejecuta en el router Dragino. El router envía los datos al puerto UART, este proceso se realizara mediante un script Lua[7]. Una vez los datos estén en el puerto UART, se conectara la placa Arduino con su módulo Datalogger al router Dragino, mediante la configuración de pines que se muestra en la Tabla 2.1.

La Tabla 2.1 muestra la conexión que se realiza entre el router Dragino y la placa Arduino[8]; los pines SIN y SOUT para el caso del router Dragino y los pines RX y TX para la placa Arduino, con esto se realiza la trasferencia de datos entre Dragino y Arduino. Para su alimentación se conectó el pin 3.3 V del router con el pin de 5 V de la placa Arduino. Se debe utilizar la misma referencia en ambos dispositivos.

Realizada la transferencia de datos, con el código creado para Arduino se procede al almacenamiento de los datos en una memoria SD con el Datalogger. Una vez realizado todo el proceso se podrá acceder a los archivos creados por el Datalogger SD y ser manipulados de la forma que se desee.

1.2. COMPILACIÓN CRUZADA.

Uno de los puntos clave del presente trabajo fue la compilación de archivos en lenguaje C en el router Dragino. Para ese fin se utilizó el proceso de compilación cruzada, el cual es la utilización de un compilador que puede crear un ejecutable para la arquitectura de otra plataforma, a la que él se ejecuta.

Dragino se basa en la arquitectura MIPS[9], esto requiere de herramientas capaces de compilar código C y crear archivos ejecutables bajo esa arquitectura.

Se utilizó la herramienta llamada Buildroot, la cual es un conjunto de archivos Makefile y de parches que hace más fácil la creación de programas que se pueden ejecutar en sistemas embebidos Linux completos.

En dichos parches se encuentra la adición de Libmodbus, haciendo que podamos crear nuestros ejecutables con la arquitectura del dispositivo embebido sin ningún inconveniente.

Al inicio de la investigación, la compilación cruzada se desarrolló con la herramienta OpenWrt SDK atheros 2.6, con la cual se pudo crear archivos ejecutables que posteriormente se utilizó en el router Dragino. Como por ejemplo el código Helloworld.c. La compilación de dicho código fue un éxito en el Dragino como se muestra en la Figura 1.2.

Figura 1. 2. Ejecución de Helloword en Router Dragino.

Se realizó otras pruebas, como crear un código que ejecutara la función seno, teniendo resultados satisfactorios.

El paso a seguir fue de compilar un código que utilizara librerías Modbus, teniendo como resultado error en la compilación; esto se debió a que dicha librería se auxilia de librerías dinámicas, como Libmodbus.so y Libmodbus.so.5. Librerías que OpenWrt SDK no puede llamar ya que no se encuentran en sus archivos Makefile.

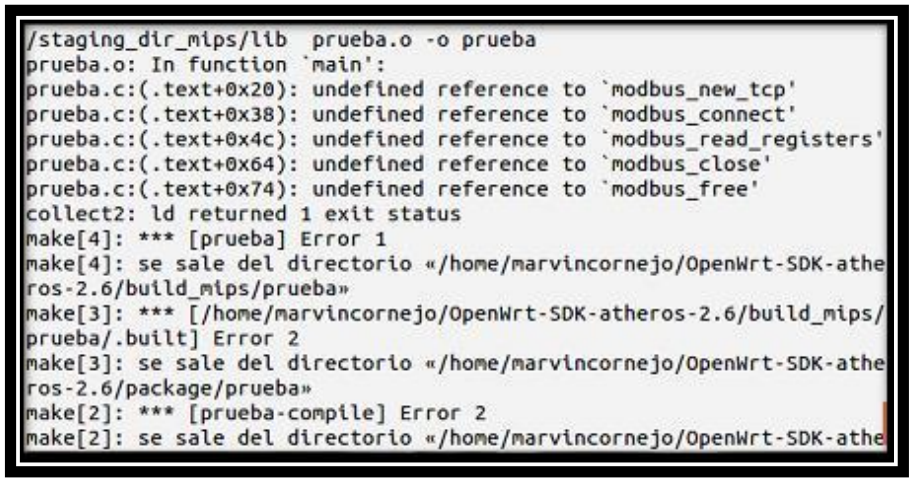


Figura 1. 3. Error al ejecutar código con librerias Modbus.

Al no tener los resultados esperados se toma la decisión de utilizar otra herramienta para la compilación, descrita al inicio de este apartado.

1.3. LENGUAJES DE PROGRAMACION UTILIZADOS.

En este proyecto se utilizó los lenguajes de programación C, Lua y el lenguaje de programación para Arduino, el cual está basado en lenguaje C y C++.

Para la ejecución del protocolo Modbus se utilizó el lenguaje de programación C. La comunicación entre Dragino y Arduino se utilizó el lenguaje Lua, este es un lenguaje de programación compacto muy popular en sistemas empotrados. Las

variables no tienen tipo, solo los datos y pueden ser lógicos, enteros, flotantes o cadenas.

Los programas Lua se escriben de forma dinámica, se ejecuta mediante la interpretación de bytecode para un equipo virtual basado en registros, y cuenta con la gestión de memoria automática [10].

El proceso es normalmente transparente al usuario y se realiza en tiempo de ejecución, pero puede hacerse con anticipación para aumentar el rendimiento y reducir el uso de memoria al prescindir del compilador.

El lenguaje Arduino está basado en los lenguajes C/C++, soporta todas las construcciones de C estándar y algunas funcionalidades de C++. Vincula las librerías de AVR libc y permite el uso de todas sus funciones.

El paquete AVR libc[11] proporciona un subconjunto de la biblioteca estándar de C para ATMEL AVR microcontroladores RISC de 8 bits. Además, la biblioteca proporciona el código de inicio básico para la mayoría de aplicaciones.

2.0. DISEÑO HARDWARE DEL PROTOTIPO.

En este capítulo se muestran las configuraciones de los diferentes tipos de hardware que conforman el prototipo.

El router Dragino interroga mediante enlaces WIFI, a cada uno de los medidores de energía instalados dentro del campus de la UES. El router pasa esta información al módulo Arduino. Este a su vez, por medio de un módulo que es capaz de leer y escribir en memorias SD, almacena dicha información.

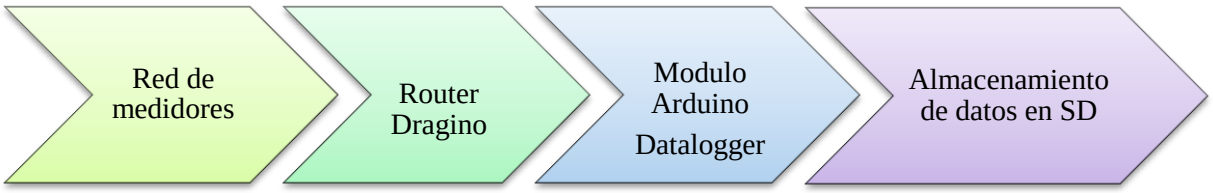


Figura 2.1. Esquema del prototipo de almacenamiento masivo de datos.

2.1. MEDIDOR SHARK Y ROUTER DRAGINO.

La comunicación entre la red de medidores y el router Dragino se basa en el protocolo Modbus. La configuración del mapa de memoria Modbus se muestra en el apéndice C.

Figura 2.2. Medidores Shark con router Dragino.

Para las pruebas de la comunicación entre los dos dispositivos se utilizó un emulador de protocolo Modbus llamado Diagslave que está configurado en modo servidor. El router Dragino corresponde a un cliente. En el apéndice D se muestra las pruebas realizadas con este emulador.

2.2. COMUNICACION ENTRE ROUTER DRAGINO Y ARDUINO.

La comunicación entre el router Dragino y la placa Arduino se realizó a través del puerto UART del Dragino y el puerto serial de la placa Arduino[12]. En el Dragino se utilizaron los pines SIN y SOUT. Estos se conectan a los pines configurados para el puerto serial de la placa Arduino los cuales son RX y TX o de otra forma los pines digital 0 y digital 1. La conexión de los dos dispositivos se muestra en la Tabla 2.1.

ARDUINO	DRAGINO
TX	SIN
RX	SOUT
5V	3.3V
GND	GND

Tabla 2.1. Conexión de pines entre Arduino y Dragino.

Es de tomar en cuenta que para tener una comunicación exitosa entre los dos dispositivos se tiene que utilizar la misma referencia. En este caso seria los puntos a tierra de los dispositivos tienen que coincidir.

La conexión entre la placa Arduino y el módulo SD se da por los pines D10(CS), D11(MOSI), D12(MISO), D13(SCK), teniendo como resultado una comunicación SPI[13] entre los dos dispositivos.

Figura 2.3. Hardware utilizado en el proyecto.

2.3. CONFIGURACION DE PUERTO UART DEL ROUTER DRAGINO.

Para la configuración del puerto UART en Dragino que viene siendo ttyS0, se tiene que tomar en cuenta ciertos requisitos de comunicación que se deben de respetar.

Lo primero es conocer donde se encuentra el dispositivo serial, como se muestra en la Figura 2.4.

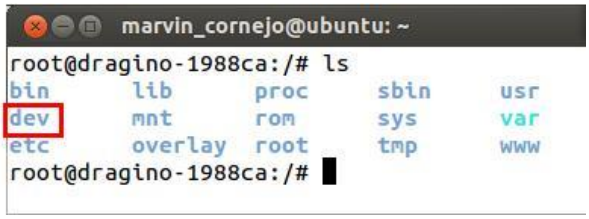


Figura 2.4. Ubicación de ttyS0.

Para conocer la configuración del puerto ttyS0, se debe modificar los permisos del puerto. Se ejecuta el siguiente comando:

```
root@dragino-1988ca:/dev# ls –la ttyS0
crw-rw-rw- 1 root          root       4,    64    Jan   1 00:00 ttyS0
root@dragino-1988ca:/dev# ls –la ttyS0
```

Figura 2.5. Permisos de ejecución para ttyS0.

Para saber a profundidad las diversas opciones con las que cuenta el puerto ttyS0 se ejecuta la siguiente línea de comando:

```
root@dragino-1988ca:/dev# stty -F /dev/ttyS0 –a
speed 9600 baud; root 24; columns 80;
intr = ^C; quit = ^\; erase = ^?; kill =^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start= ^Q; stop = ^S; susp = ^Z;
rprnt = ^R; werase= ^W; lnext = ^V; vflush = ^O; min = 1; time = 0;
-parenb –parodd –cs8 hupcl –cstopb cread clocal –crtscts
-ignbrk –brkint –ignpar –parmrk –inpck –istrip –inlcr –igncr icrnl
ixon –ixoff –iuclc –ixany –imaxbel
Opost –olcuc –ocrnl onlcr –onocr –onlret .ofill –ofdel nl0 cr0 tab0
bs0 vt0 ff0 isig icanon iexten echo echoe echok –echonl –noflsh –
xcase –tostop –echoprt echoctl echoke
```

Figura 2.6. Opciones de ttyS0.

.

El router Dragino y la placa Arduino tienen que estar configurados en la misma velocidad en baudios, se cambia dicho parámetro.

```
root@dragino-1988ca:/dev# stty -F /dev/ttyS0 4800
root@dragino-1988ca:/dev# stty -F /dev/ttyS0
speed 4800 baud;
intr = ^C; quit = ^\; erase = ^?; kill =^U; eof = ^D; eol =
<undef>;
eol2 = <undef>; swtch = <undef>; start= ^Q; stop = ^S; susp = ^Z;
rprnt = ^R; werase= ^W; lnext = ^V; vflush = ^O; min = 1; time = 0;
-brkint –imaxbel

root@dragino-1988ca:/dev# stty -F /dev/ttyS0 9600
root@dragino-1988ca:/dev# stty -F /dev/ttyS0
speed 9600 baud;
intr = ^C; quit = ^\; erase = ^?; kill =^U; eof = ^D; eol =
<undef>;
eol2 = <undef>; swtch = <undef>; start= ^Q; stop = ^S; susp = ^Z;
rprnt = ^R; werase= ^W; lnext = ^V; vflush = ^O; min = 1; time = 0;
-brkint –imaxbel
```

Figura 2.7. Cambio de baud rate para ttyS0.

Después de haber configurado el puerto ttyS0 a la velocidad de 9600 baud, el router Dragino está listo para poder comunicarse con la placa Arduino, que también se configura a esa velocidad.

2.4. PLACA ARDUINO CON DATALOGGER SD.

Una vez configurado el puerto serial del router Dragino, para su acoplamiento con la placa Arduino, se da paso a la conexión del módulo Datalogger SD. Para realizar dicha conexión se tiene que conocer sobre el protocolo SPI[14] con el que se comunican la placa Arduino y el módulo Datalogger SD.

Para una conexión SPI siempre hay un dispositivo maestro, que en este caso sería la placa Arduino, que controla los dispositivos periféricos, en este caso el Datalogger SD.

Normalmente hay cuatro líneas comunes a todos los dispositivos:

MISO (Master In Slave Out) - La línea de esclavos para el envío de datos al maestro,

MOSI (Master Out Slave In) - La línea principal para el envío de datos a los periféricos.

SCK (Serial Clock) - Los impulsos de reloj que sincroniza la transmisión de datos generada por el maestro y una línea específica para cada dispositivo.

SS (Slave Select) - el pasador en cada dispositivo que el maestro puede utilizar para activar y desactivar dispositivos específicos.

Cuando en un dispositivo el pin **Slave Select** tiene un estado lógico bajo este se comunica con el maestro. Cuando el estado lógico es alto, simplemente ignora al maestro. Esto le permite tener múltiples dispositivos SPI que comparten la misma MISO, MOSI, y las líneas de CLK. Para el caso del proyecto este pin se fijó en baja, ya que la placa Arduino dispone de un único periférico.

Placa Arduino	MOSI	MISO	SCK	SS(esclavo)	SS(maestro)
Uno o Duemilanove	11 o ICSP-4	12 o ICSP-3	13 o ICSP-3	10	-
Mega 1280 o mega 2560	51 0 ICSP-4	50 o ICSP-1	52 o ICSP-3	53	-
Leonardo	ICSP-4	ICSP-1	ICSP-3	-	-
Debido	ICSP-4	ICSP-1	ICSP-3	-	4, 10,52

Tabla 2.2. Conexión SPI.

3.0. CONFIGURACIÓN SOFTWARE.

En este capítulo se explica la configuración software de los diferentes componentes que conforman el prototipo. Se inicia con la explicación de cómo se realiza la compilación cruzada, luego se explica las herramientas para el router Dragino y la placa Arduino.

Para la ejecución del protocolo Modbus en el router Dragino se hizo uso de dos herramientas para realizar la compilación cruzada, la primera OpenWrt SDK 2.6, con la cual no se obtuvieron los resultados esperados y la segunda fue la herramienta Buildroot, que se utilizó al final del proyecto por sus resultados satisfactorios.

Como resultado se tiene el almacenamiento en una memoria SD de los datos recibidos por medio del módulo Datalogger SD de Arduino.

3.1. CONFIGURACION DE PC PARA BUILDROOT.

Buildroot es una herramienta simple para crear archivos ejecutables (binarios) para sistemas Linux embebidos, la librería Buildroot 2011.11 cuenta con la particularidad de tener entre sus parches Libmodbus[15].

En la computadora que se utilizó para la compilación cruzada disponía del sistema Ubuntu 12.04.

El compilador instalado en el sistema Ubuntu 12.04 es GCC 4.3, muy importante es que debe de coincidir con el que se utiliza en Buildroot, para evitar errores de compatibilidad.

La Tabla 3.1 muestra los paquetes necesarios para la instalación de Buildroot.

PAQUETES	INSTALACION
Build-essential	sudo apt-get install build-essential
subversion	sudo apt-get install subversion
zlibb1g-dev	sudo apt-get install zlibb1g-dev
git-core.	sudo apt-get install git-core
libssl-dev.	sudo apt-get install libssl-dev
xmlto.	sudo apt-get install xmlto
gawk.	sudo apt-get install gawk
texinfo	sudo apt-get install texinfo
bison	sudo apt-get install bison

Tabla 3.1. Paquetes para instalación de Buildroot en pc.

Una vez instalados los paquetes mostrados en la Tabla 3.1, se procede a la instalación de Buidroot 2011.11.

Se obtiene la distribución de Buildroot 2011.11 dela siguientes sitios:

- http://buildroot.net/downloads/buildroot-2011.11.tar.gz
- http://buildroot.net/downloads/buildroot-2011.11.tar.bz2

Una vez descargado el archivo se procede a descomprimirlo desde una terminal con la siguiente línea de comando: **tar xvjf buildroot-2011.11.tar.gz.**

Se creó el directorio /opt dentro del cual está el subdirectorio con el nombre de /toolchain-mips dentro de este se copia los archivos para la instalación de Buildroot 2011.11.

Como lo muestra la Figura 3.1, en una terminal se escribe la siguiente línea:

```
marvin_cornejo@ubuntu:~/opt/toolchains-mips/buildroot-2011.11$ make menuconfig
```

Figura 3.1. Comando para ingresar a menu de configuracion de Buildroot.

Una vez ejecutado el comando de la Figura.3.1, se procede a ingresar al menú de configuración de Buildroot, véase la Figura 3.2.

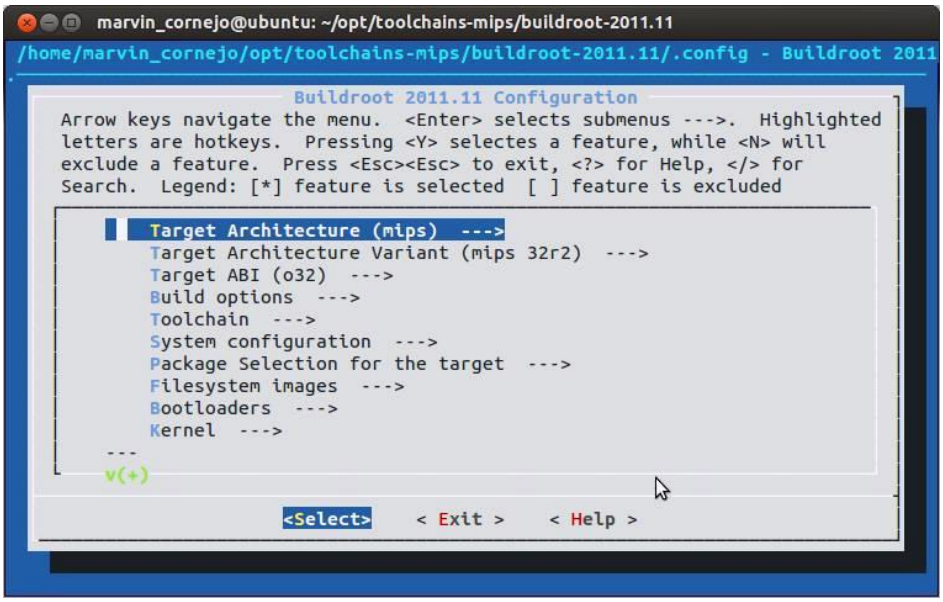


Figura 3.2. Menú de configuración de Buildroot.

Como se puede observar de la Figura 3.2, el menú de configuración cuenta con un submenú de opciones para la configuración de la compilación cruzada, teniendo en cuenta los requisitos siguientes:

- la arquitectura del sistema embebido, para el router Dragino es Mips 32.
- Los Toolchain, los cuales son las herramientas que se utilizan para la compilación, como el tipo de librerías gcc, para el caso gcc 4.3, ya que con este cuenta el router.
- Kernel Headers, el cual se utiliza linux 3.1.x kernel Headers.
- En la configuración del sistema, se selecciona la velocidad de baudios a la cual trabaja el router, para el caso 9600 baud.
- En la selección de paquetes para la tarjeta del sistema, se accede al submenú de librerías, posteriormente Networking y como se observa en la Figura 3.3. se selecciona la librería Libmodbus.

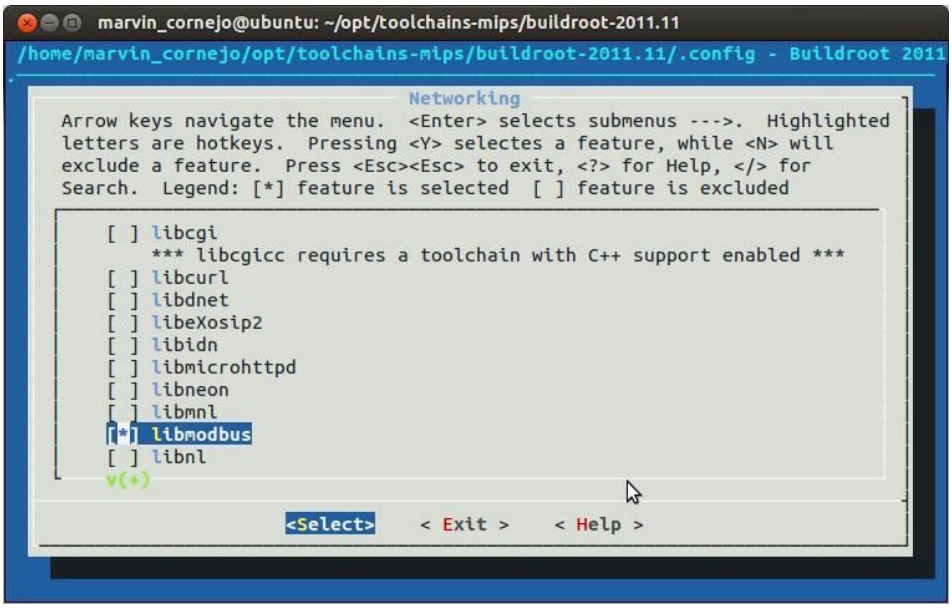


Figura 3.3. Configuración de Libmodbus en Buildroot.

Una vez terminada la configuración de Buildroot, se procede a guardar el proceso como se muestra en la Figura 3.4.

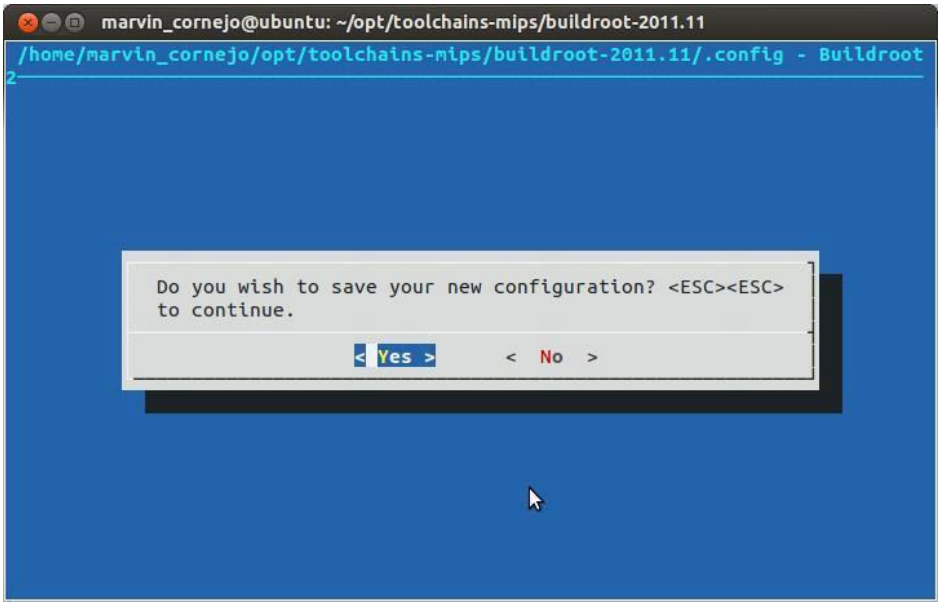


Figura 3.4. Guardando configuración de Buildroot.

Realizado el paso anterior, se procede a ejecutar la siguiente línea de comando:

```
Marvin_cornejo@ubuntu:~/opt/toolchains-mips/buildroot-2011.11$ make
```

Figura 3.5. Línea para poder realizar la configuración de Buildroot.

Es de mencionar que cuando se realizó el proceso de configuración e instalación se tiene que contar con tiempo para dicho proceso ya que depende de la velocidad con que cuenta la pc y la conexión a internet, ya que varios paquetes utilizados son descargados en dicho proceso.

Para finalizar se debe configurar las rutas para poder hacer que el compilador GCC instalado en la pc pueda entrelazar los compiladores creados con las características de los sistemas embebidos, como se muestra en la Figura 3.6.3.3.

```
export TOOLCHAINS_MIPS=/opt/toolchains-mips/buildroot-2011.05
export PATH=$TOOLCHAINS_MIPS/output/host/usr/bin:$PATH
export AR=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-ar
export AS=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-as
export CC=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-gcc
export CPP=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-cpp
export CXX=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-g++
export LD=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-ld
export GCC=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-gcc
export NM=$TOOLCHAINS_MIPS/output/host/usr/bin/mips-linux-nm
```

Figura 3.6. Exportación de rutas para gcc.

3.2. COMPILACION CRUZADA CON OPENWRT SDK.

Dragino trae por defecto el sistema Linux OpenWrt Backfire 10.03.1 Atheros, el cual se puede descargar desde la siguiente dirección: http://downloads.openwrt.org/backfire/10.03.1/ o desde la propia wiki de Dragino http://www.dragino.com/downloads/.

Al revisar los paquetes que contiene OpenWrt de Dragino, no cuenta con compiladores para programas en C y C++.

Es de mencionar que la herramienta OpenWrt SDK que se describe en este apartado no es el método con el cual se trabajó para la compilación cruzada, por los inconvenientes que se describen más adelante, pero se menciona por su utilidad en códigos menos complejos que los utilizados en la investigación.

Después de descargar OpenWrt SDK, se procede a descomprimir e instalar como lo muestra la Figura 3.7.

```
Marvin_cornejo@ubuntu:~$ tar zxvf OpenWrt-SDK-atheros-2.6-for-Linux-
i686-UCB.tar.bz2
```

Figura 3.7. Descomprimir OPenWrt SDK

Al abrir la carpeta OpenWrt-SDK-Atheros-2.6 se podrán observar las diferentes carpetas:

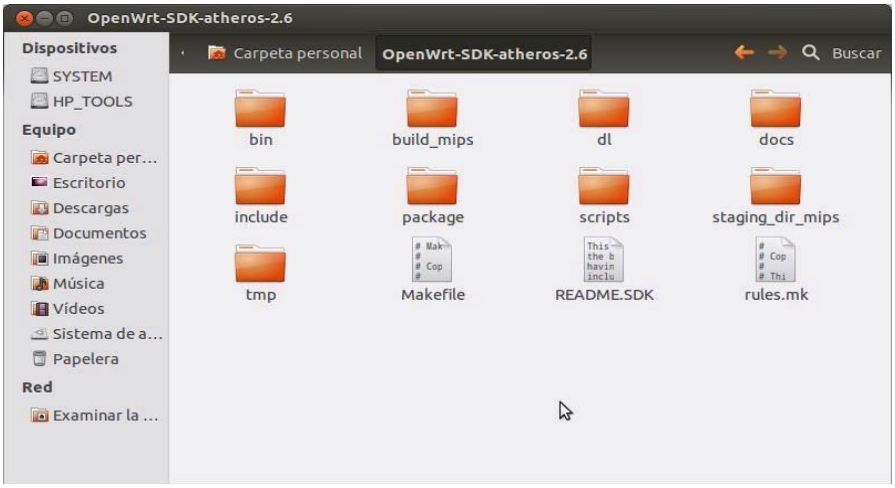


Figura 3.8. Carpetas de OpenWrt SDK atheros 2.6.

Configuración de ficheros Makefile para compilación Cruzada.

Los makefiles son los ficheros de texto que utiliza Make para llevar la gestión de la compilación de programas. Se podrían entender como los guiones de la película que quiere hacer Make, o la base de datos que informa sobre las dependencias entre las diferentes partes de un proyecto. Todos los Makefiles están ordenados en forma de reglas, especificando que es lo que hay que hacer para obtener un módulo en concreto.

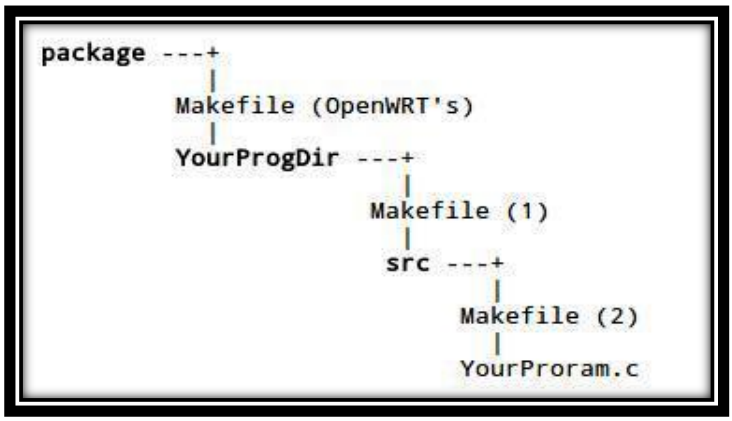


Figura 3.9. Esquema del orden de los Makefile.

Según el esquema que se muestra en la Figura 3.10, makefile(1) se tiene que modificar ciertos parámetros, que tiene que ver con el nombre del archivo c que se compilará, como se muestra a continuación:

```
Include $(TOPDIR)/rules.mk
# Name and reléase number of package
PKG_NAME:=helloword
PKG_RELEASE:=1
PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)
Include $(INCLUDE_DIR)/package.mk
Define package/helloword
      SECTION:=utils
      CATEGORY:=Utilities
      TITLE:= helloword – prints a snarky message
Endef
Define package/helloword/description
      If you can´t figure out what this program does, you´re probably
      Brain-dead and need immediate medical attention.
Endef
Define Build/Pepare
      Mkdir –p $(PKG_BUILD_DIR)
      $(CP) ./src/* $(PKG_BUILD_DIR)/
Endef
Define Package/helloword/install
      #$(INSTALL_DIR) $(1)/bin
      #$(INSTALL_BIN) $(PKG_BUILD_DIR)/helloword $(1)/bin/
      Cp $(PKG_BUILD_DIR)/helloword /temp
Endef
$(eval $(call Buildpackage, helloword))
```

Figura 3.10. Configuración de Makefile(1).

Los parámetros donde se introduce el nombre del archivo c son los que se modifican para poder compilar otros archivos, este Makefile es tanto para C como para C++.

El siguiente Makefile es el que hace la compilación del archivo c y C++, es de observar que para archivos c se utiliza el prefijo **(CC)** y para archivos C++ **(CXX).**

```
Helloword: helloword.o
      $(CC) $(LDFLAGS) helloword.o –o helloword
      Helloword.o: helloword.c
      $(CC) $(CFLAGS) –c helloword.c
# remove object files an executable when user ececutes “make clean”
      Rm *.o helloword
```

Figura 3.11. Makefile para archivo C.

Para compilar otro archivo C se tiene que cambiar el nombre del archivo actual en el Makefile al nuevo que se quiere compilar. Lo mismo se realiza para la compilación de archivos C++, como se muestra en la siguiente Figura 3.12:

```
Helloword: helloword.o
      $(CXX) $(LDFLAGS) helloword.o –o helloword
      Helloword.o: helloword.c
      $(CXX) $(CXXFLAGS) –c helloword.c
# remove object files an executable when user ececutes “make clean”
      Rm *.o helloword
```

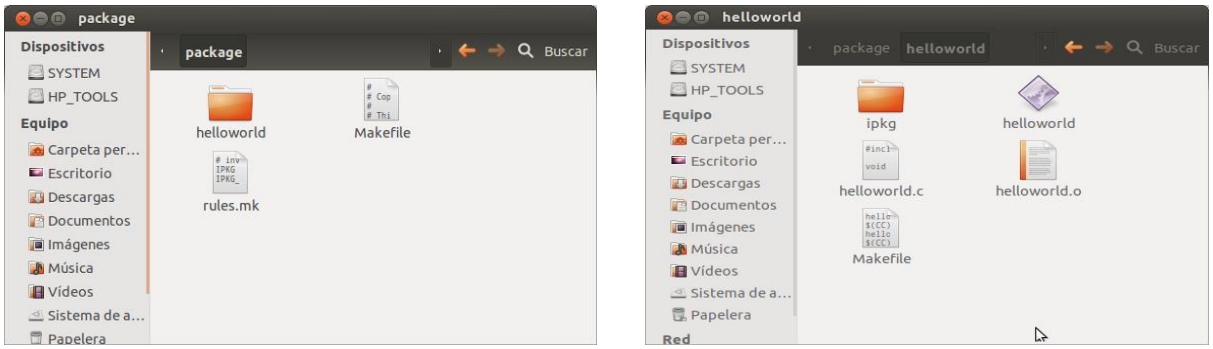
Figura 3.12. Makefile para archivo C++.

Al completar todos los pasos anteriores, se procede a ejecutar desde una Terminal el fichero Makefile principal el cual se encuentra en la carpeta de OpenWrt-Atheros-2.6; este es el encargado de ejecutar las reglas de los demás Makefiles que se utilizan para la compilación de los archivos c y c++. Como se muestra en la Figura 3.13.

```
Marvin_cornejo@ubuntu:~/OpenWrt-SDK-atheros-2.6$ make V=99
Make package/compile
Make[1]: se ingresa al directorio “/home/marvin_cornejo/OpenWrt-SDK-atheros-2.6”
Make[2]: se ingresa al directorio “/home/marvin_cornejo/OpenWrt-SDK-atheros-2.6”
Collecting package info:package/helloword/home/marvin_cornejo/OpenWrt-SDK-atheros-
2.6/include
Host:8: /home/marvin_cornejo/OpenWrt-SDK-atheros-2.6/tmp/.host.mk:
Collecting package info:done
Make[3]: se sale del directorio directorio “/home/marvin_cornejo/OpenWrt-SDK-
atheros-2.6”
Make[2]: se sale del directorio directorio “/home/marvin_cornejo/OpenWrt-SDK-
atheros-2.6”
Make –C package compile SDK=1
```

Figura 3.13. Ejecución de Makefile para compilación cruzada.

Se tiene acceso a la carpeta package, en la cual se almacenara un fichero Makefile el cual será el que da las especificaciones para la ejecución de la compilación cruzada. Se tiene una subcarpeta con el nombre del código a crear, en la cual se tiene el código C/C++ y su respectivo Makefile el cual es el encargado de compilar el archivo, como se muestra en la siguiente figura:



a) **b)**

Figura 3.14. a) Carpeta helloworld del código C/C++. b) archivos generados al compilar código heloworld.c/cpp.

Como se observa en la Figura 3.14 (b) se generan los archivos helloworld.o, helloworld.bin y la carpeta ipkg. Todo esto es creado por los parámetros especificados en los ficheros Makefile.

Al finalizar se obtiene los ejecutables binarios e instalables ipk, con esto se procede a copiarlos en el router Dragino.

Como ya se dijo al inicio de esta sección la compilación cruzada con esta herramienta no fue satisfactoria para el proyecto. Se observó al momento de ejecutar los script bajo el protocolo Modbus, que presentaba un error en la compilación, y de ahí el hecho de no poder crearse el ejecutable ni el archivo ipk.

El problema se generaba al momento que OpenWrt SDK intentaba enlazar las librerías dinámicas que utiliza Libmodbus, como libmodbus.so, libmodbus.so.5.

Esto se debe a que dentro de las líneas de compilación de la Figura 3.10, no se le indican que el código C, tiene librerías adicionales, es por ello que solo genera los ejecutables de archivos que no necesitan librerías dinámicas.

Por ese motivo, se desistió de trabajar con dicha herramienta y se buscó otra opción.

3.3. COMPILACION CRUZADA CON BUILDROOT.

Después de haber instalado adecuadamente las herramientas de la seccion 3.1 se procede a compilar.

Para realizar la compilación cruzada con la herramienta Buildroot, se ejecuta la siguiente instrucción que muestra la Figura 3.7.

```
marvin_cornejo@ubuntu:~/Escritorio/tesis_marvin/Dragino$
‘/home/marvin_cornejo/opt/toolchains-mips/buildroot-2011.11/host/usr/bin/mips-
linux-gcc’ medidor1.c –o medidor1 `pkg-config –libs –cflags libmodbus`
```

Figura 3.15. Compilación cruzada con Buildroot.

La Figura 3.15 detalla la forma de compilar con Buildroot. En el directorio /host/usr/bin/ se encuentran los compiladores de Buildroot, para el caso se utilizó mips-linux-gcc, ya que el código está escrito en C. si en el caso estuviera escrito en C++, el compilador seria mips-linux-g++.

La línea de instrucción para compilar un archivo Modbus es la misma que se utiliza para compilar un archivo Modbus en una computadora.

Buildroot utiliza el compilador ya instalado en la computadora, simplemente crea un enlace con el compilador creado con las características del router Dragino, es decir

con la arquitectura MIPS. Una vez creado el ejecutable se procede a copiarlo al router.

3.4. LIBMODBUS EN ROUTER DRAGINO.

Una de las tareas más difíciles de superar fue dotar al router Dragino de la capacidad de ejecutar archivos en lenguaje C. Una vez superado este punto se procedió a la configuración que debía tener Libmodbus en el router Dragino, se tomó de referencia el archivo libmodbus.la, el cual indica donde se ubican las librerías dinámicas [16] que utiliza el protocolo. Como lo muestra la Figura 3.18.

```
# linker flags that can not go in dependency_libs.
Inherited_linker_flags= ‘’
# Names of this library.
Library_names = ‘libmodbus.so.5.0.2 libmodbus.so.5 libmodbus.so’
#version information for libmodbus.
Current=5
Age00
Revision=2
# is this an already installed library?
Installed=yes
Shouldnotlink=no
# files to dlopen/dlpreopen
Dlopen=’’
Dlpropen=’’
# directory that this library needs to be installed in:
Libdir=’/usr/local/lib’
```

Figura 3.16. Parte del script libmodbus.la.

La última línea del código muestra donde se guardaran los script libmodbus.so.5.0.2, libmodbus.so.5 y libmodbus.so.

Se crean directorios /local/lib en el router Dragino y se copia los script libmodbus.so.5.0.2, libmodbus.so.5 y libmodbus.so.

Se realiza una copia del script libmodbus.so.5 en la siguiente dirección **/usr/lib/,** donde se encuentran todas las librerías dinámicas de los paquetes instalados en el sistema OpenWrt.

El archivo libmodbus.so.5 tiene que ser el que se encuentra en los directorios de configuración de Buildroot localizado en buildroot2011.11/output/usr/package/lib/libmodbus3.0.3.

Figura 3.17.Ubicación de libmodbus.so.5 en router Dragino.

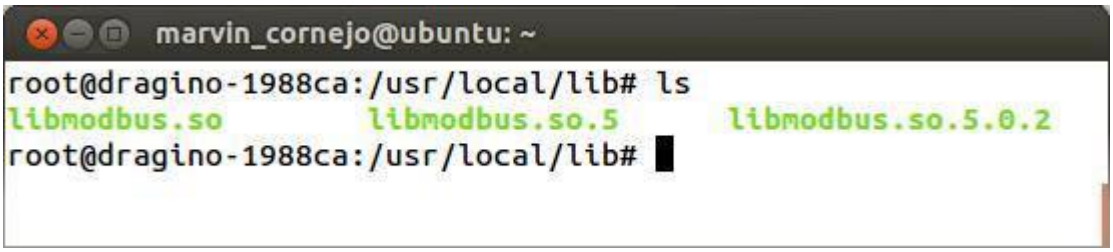


Figura 3.18. Script para ejecución de protocolo Modbus en Dragino.

Una vez realizada la copia de los script en sus directorios correspondientes, como lo muestran la Figura.3.17 y Figura.3.18. El router Dragino ya está capacitado para la ejecución del protocolo Modbus.

El router Dragino debe contar con las librerías de C, que para el caso se utiliza la librería uClib [17] que se encuentran en el paquete libgcc_4.3.3+cs-43.32_atheros.ipk. El cual se descarga desde la siguiente dirección.

http://downloads.openwrt.org/backfire/10.03.1/atheros/packages/.

3.4. CODIGO LUA.

En Lua, se creó un pequeño script, el cual tiene la capacidad de poder ejecutar el archivo.bin obtenido de la compilación cruzada, y este a su vez poder tomar en formato de cadena de caracteres las respuesta enviada por el protocolo Modbus mediante un comando print.

La lectura de los datos enviados por el protocolo Modbus, se realizó mediante la creación de una función como se muestra en la Figura.3.19.

```
nixio = require 'nixio'          --biblioteca para Puerto serial
function DataToUart(message)     --funcion para la escritura en
ttys0
     serialout=nixio.open("/dev/ttyS0","W") –se abre interfaz
UART
     men=string.format("%8s",message)–el valor leído se pasa a
formato String
     aa=serialout:write(men)     --se escribe en el puerto UART
     serialout:close()           --se cierra la interfaz UART
end                              --fin de la función
msg=(string.gsub(io.popen("./sensor01"):read("*a"),"%s+$",""))
--se lee el dato enviado por el ejecutable
DataToUART(msg) --se envía la cadena de caracteres a puerto
UART
```

Figura 3.19. Código write_uart.lua.

Parámetros utilizados.

- **Require:** se utiliza para llamar a una biblioteca para Lua.
- **Nixio:** biblioteca Lua para entrada/salida de datos en un puerto.
- **String.format:** devuelve una versión formateada de sus argumentos, siguiendo la descripción dada en su primer argumento, que en nuestro caso es una cadena de caracteres(%s), se puede utilizar otros tipos de formatos como para enteros(%d), flotantes(%f), etc.
- **string.gsub:** devuelve una copia de s (para el caso se usó popen), y lo reemplaza con una cadena de caracteres.
- **io.popen:** comienza a ejecutar el programa en un proceso separado y retorna un descriptor de ficheros el cual se usa para leer o escribir datos, en nuestro caso se utilizó para la lectura de datos.

3.5. CODIGO ARDUINO.

La placa Arduino recoge del router Dragino los datos correspondientes a mediciones eléctricas. La placa Arduino guarda esta información en un módulo Datalogger para memoria SD. Los datos se guardan en archivos.

Para la utilización del módulo datalogger SD, se hace uso de la librería que configura la creación y almacenamiento de datos en la SD. Dicha librería está definida como SD.h, la cual permite la lectura y escritura en la memoria SD [18].

Alguna de las funciones que se utiliza para la configuración de la tarjeta SD y poder manipular sus archivos son:

Función.	*Descripción.*
SD.begin()	Inicializa la biblioteca SD y la tarjeta.
SD.open()	Abre un archivo en l tarjeta SD, si se abre para la escritura y si no existe el archivo lo creara.
File.flush()	Asegura que ningun byte escrito en el archivo se guarde en la tarjeta.
pritnt()	Imprime los datos en los archivos abiertos para la escritura.
println()	Imprime los datos seguida de un avance de linea y retorno de carro.
File.read()	Lee un byte del archivo.
File.write()	Escribir datos en el archivo
File.close()	Cierra el archivo y se asegura que todos los datos se han guardado en a tarjeta SD.

Tabla 3.2. Funciones utilizadas por SD.h

En el Arduino se divide el código en dos funciones, la función Setup y la función Loop.

La función Setup es donde se inicializan las variables, las librerías y procesos, se declara alguna funcionalidad de los pines. Tiene la particularidad de que se ejecuta una sola vez al conectarse la placa o presionar reset.

La función loop como su nombre lo describe, genera un lazo de ejecución continuamente, permitiendo al programa variar y responder. Se utiliza siempre y cuando este la función setup ya que contiene las variables inicializadas.

Para la implementación del código Arduino se tiene en cuenta que los datos serán recibidos desde el puerto serial, los pines D0(RX) y D1(TX), que estarán conectados al puerto UART del router Dragino.

Los datos recibidos son cadena de caracteres, es por ello que se utiliza la función String, la cual nos deja manipular caracteres. Estos son leídos por Serial.read(), los cuales serán pasados a una variable tipo caracter y este a su vez una string, para

evitar que Arduino capte el valor como ASCII, ya que es un error común en dicho sistema.

Figura 3.20. Conexión Dragino-Arduino.

Una vez realizado estos pasos se procede a escribir los resultados en la memoria SD, como se muestra en el fragmento de código de la Figura.3.21.

```
//iniciando tarjeta SD
     Serial.print("iniciando tarjeta SD …");
//se asegura que el pin del chip seect este como salida (no se usa)
     pinMode(10,OUTPUT);
//ver si la tarjeta está presente e instalada
     If (!SD.begin(chipSelect)) {
     Error("la memoria está dañada o no está presente …");
}
If (!SD.exists(filename))
     {
     //solamente abre un nuevo archivo si este no existe
     ArchivoSD=SD.open(filename, FILE_WRITE);
     break; // dejar el lazo
}
```

Figura 3.21. Fragmento de código Arduino para almacenar datos en SD.

Si no se encuentra presente la memoria SD en el módulo Datalogger, no se realiza ningún proceso dado que se ha configurado para que se active siempre y cuando esté presente la memoria SD.

En la función Loop se ejecuta la lectura del puerto UART del router dragino, los datos son leídos por la función Serial.read(), la cual se almacena en una variable tipo byte, ya que es el tipo de dato que envía UART, y este se pasa a un tipo carácter para luego ser enviado a una cadena de caracteres con String, para luego ser guardadas en la memoria SD.

```
void loop(void)
{
//Para guardar en la memoria SD
archivoSD = SD.open(filename, FILE_WRITE);

   // envio de datos solo cuando se receive datos:
  if (Serial.available()){

// envio de datos desde UART Dragino a puerto serial Arduino
   while (Serial.available()>0){

      numChar=Serial.read();
      incomingbyte=numChar;
      recive=String(incomingbyte);
      Serial.print(recive);
      archivoSD.print(recive); //se manda el valor del estado medido
      //delay(8000);
    }

  Serial.println();
  Serial.flush();
  archivoSD.println();
 }
```

Figura 3.22. Función loop para recepción de datos desde UART Dragino.

El almacenamiento de los datos obtenidos del puerto se realiza creando o abriendo el archivo en modo de escritura, estando esté listo para la escritura. Se utiliza la línea archivo.print() la cual se encarga de copiar los datos al fichero que hemos designado como destino. Realizado todos estos pasos se cumple con el objetivo de poder almacenar los datos proporcionados por el script Modbus, en una memoria SD con la utilización de una placa Arduino y un módulo Datalogger SD.

4.0. COMUNICACION ENTRE DISPOSITIVO DE ALMACENAMIENTO Y RED DE MEDIDORES DE LA UES.

En este capítulo se muestra la configuración de red que se realiza el router Dragino para poder tener acceso a la red de medidores de la UES. Se configura el router Dragino como un súper nodo, para poder interrogar a la mayoría de medidores instalados.

Se utiliza el protocolo BATMAN el cual es el encargado de enrutar los datos de una red malla hasta su destino correcto. El cual está disponible su distribución para el router Dragino.

Al final se muestra la interacción que hay entre el dispositivo de almacenamiento masivo, la interrogación y adquisición de datos de la red de medidores de la Universidad de El Salvador.

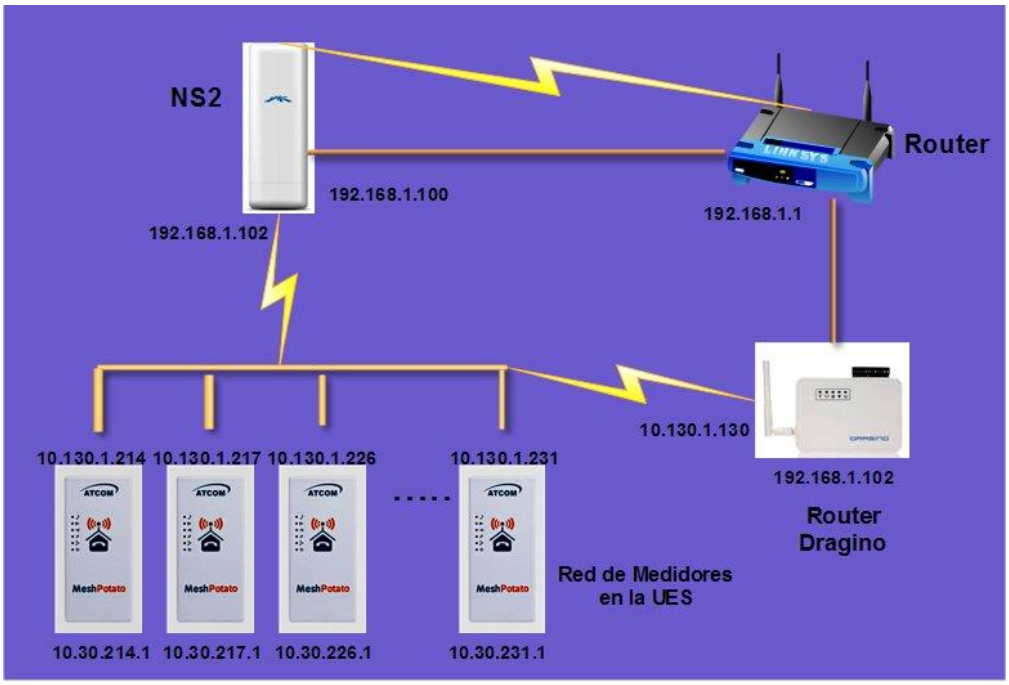


Figura 4.1. Topología de conexión de router Dragino y red de medidores

4.1. INSTALACION DE BATMAND.

Para realizar la instalación de BATMAN. Se descargan los siguientes archivos que son necesarios para el funcionamiento de dicho protocolo.

- librt_0.9.30.1-43.32_atheros.ipk
- libpthread_0.9.30.1-43.32_atheros.ipk
- kmod-tun_2.6.30.10-1_atheros.ipk
- batmand_r1439-1_atheros.ipk
- vis_r1439-1_atheros.ipk.

Se descargan del siguiente sitio:

http://downloads.openwrt.org/backfire/10.03.1/atheros/packages/

Se copian todos los archivos al router como se muestra en la Figura 4.2.

```
Marvin_cornejo@ubuntu:~$ scp batmand_r1439-1_atheros.ipk root@192.168.255.1:/
Marvin_cornejo@ubuntu:~$ scp librt_0.9.30.1-43.32_atheros.ipk root@192.168.255.1:/
Marvin_cornejo@ubuntu:~$ scp libpthread_0.9.30.1-43.32_atheros.ipk root@192.168.255.1:/
Marvin_cornejo@ubuntu:~$ scp kmod-tun_2.6.30.10-1_atheros.ipk root@192.168.255.1:/
Marvin_cornejo@ubuntu:~$ scp vis_r1439-1_atheros.ipk root@192.168.255.1:/
```

Figura 4.2. Ejemplo de cómo copiar los archivos al router Dragino.

Terminado el proceso de copia de los archivos anteriormente mencionados se procede a la instalación.

Se configura la IP de la PC en el rango al cual pertenece el router Dragino con la siguiente línea: **Ifconfig eth0 192.168.103/24 up.**

Ingresamos al router Dragino con la siguiente línea **ssh root@192.168.1.102** y se instalan los archivos con la línea que muestra la Figura 4.3.

.

```
root@192.168.1.102:~$ opkg install librt_0.9.30.1-43.32_atheros.ipk
root@192.168.1.102:~$ opkg install libpthread_0.9.30.1-43.32_atheros.ipk
root@192.168.1.102:~$ opkg install kmod-tun_2.6.30.10-1_atheros.ipk
root@192.168.1.102:~$ opkg install batmand_r1439-1_atheros.ipk
root@192.168.1.102:~$ opkg install vis_r1439-1_atheros.ipk.
```

Figura 4.3. Instalación de archivos en router Dragino.

Habiendo terminado la instalación de todos los paquetes, se procede a habilitar batmand y vis como se muestra en la Figura 4.4.

```
root@192.168.1.102:~$ /etc/init.d/batmand enable
root@192.168.1.102:~$ /etc/init.d/vis enable
```

Figura 4.4. batmand y vis habilitados.

Ahora se borran la siguientes IP-TABLES:

```
root@192.168.1.102:~$ uci commit
root@192.168.1.102:~$ cd /etc/modules.d/
root@192.168.1.102:/etc/modules.d$ ls
20-crc-ccitt      30-tun      40-pppoe          42-ipt-nat  50-madwifi
30-ppp            40-ipt-core 41-ipt-conntrack 45-ipt-nathelper
root@192.168.1.102:/etc/modules.d$ rm –rf 40-ipt-core
root@192.168.1.102:/etc/modules.d$ rm –rf 41-ipt-conntrack
root@192.168.1.102:/etc/modules.d$ rm –rf 42-ipt-nat
root@192.168.1.102:/etc/modules.d$ rm –rf 45-ipt-nathelper
```

Figura 4.5. Borrado de IP TABLES

Se reinicia el sistema:

```
root@192.168.1.102:~$ reboot; exit
```

Figura 4.6. Reinicio del sistema.

4.2. CONFIGURACIÓN DE ROUTER DRAGINO COMO SUPER NODO.

Para realizar la configuración del router como súper nodo[19] se tienen que modificar los siguientes archivos:

- Wireless
- Network
- Batmand

En el router Dragino, se tiene acceso a la siguiente dirección /etc/config. Donde se encuentran los archivos antes mencionados. Se procede a modificar el archivo wireless, quedando de la forma como se muestra en la Figura 4.7.

```
root@192.168.1.102:~$/etc/config# cat wireless
config 'wifi-device'    'wifi0'
    option 'type'       'atheros'
    option 'channel'    '11'

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

Figura 4.7. Archivo de configuración wireless.

En archivo de configuración wireless, se modifican los términos como el tipo de distribución del OS, el canal en la cual está configurada la red de medidores, que para este caso es el 11.
En la configuración del archivo network, es de observar que se modifica la IP para el ingreso al router por ssh, esto debido a que tiene que estar en el rango de IP de los medidores de energía.

```
root@192.168.1.102:~$/etc/config# cat network
config 'interface' 'loopback'
    option 'ifname'      'lo'
    option 'proto'       'static'
    option 'ipaddr'      '127.0.0.1'
    option 'netmask'     '255.0.0.0'

config 'interface' 'lan'
    option 'ifname'      'eth0'
    option 'proto'       'static'
    option 'ipaddr'      '192.168.1.102'
    option 'netmask'     '255.255.255.0'
    option 'gateway'     '192.168.1.1'
    option 'dns'         '192.168.1.1'

config alias
    option 'interface'    'lan'
    option 'proto'        'static'
    option 'ipaddr'       '10.30.1.1'
    option 'netmask'      '255.255.255.0'

config 'interface' 'wifi0'
    option 'ifname'        'ath0'
    option 'proto'         'static'
    option 'ipaddr'        '10.130.1.1'
    option 'netmask'       '255.255.255.0'
```

Figura 4.8. Archivo de configuración network.

Se crea una interfaz “alias” para tener acceso a la red de medidores, para luego poder interrogarlos sobre el funcionamiento de dicha red.

La configuración de la interfaz “wifi0”, permite la comunicación vía WIFI entre los nodos en la red de medidores. El router Dragino cuenta con la capacidad de poder interrogar a la mayoría de los nodos de dicha red.

Para la configuración del archivo batmand en el router Dragino se toma de referencia el instalado en el súper nodo NS2, de la red de medidores de la UES.

```
root@192.168.1.102:~$/etc/config# cat batmand
config 'batmand' 'general'
    option 'interface' 'ath0'
    option 'hna' '10.30.1.0/24'
    option 'originator_interval' ''
    option 'preferred_gateway' ''
    option 'policy_routing_script' ''
    option 'disable_client_nat' ''
    option 'disable_aggregation' ''
    option 'gateway_class' '5000'
    option 'routing_class' ''
    option 'visualisation_srv' '10.130.1.1'
```

Figura 4.9. Archivo de configuración batmand.

Una vez realizado todos los cambios en los tres archivos de configuración, se reinicia el router y se vuelve a tener acceso a él por medio de ssh.

Ejecutamos la línea **ps –x** para observar los procesos que el router tiene en ejecución, se observa que tanto batmand como vis trabajan en el sistema.

```
root@192.168.1.102:~$ ps -x
877 root 1016 S batmand -a 10.30.1.0/24 -g 5000 -s 10.130.1.1 ath0
878 root 1016 S batmand -a 10.30.1.0/24 -g 5000 -s 10.130.1.1 ath0
880 root 1016 R batmand -a 10.30.1.0/24 -g 5000 -s 10.130.1.1 ath0
890 root 940 S vis -j ath0 eth0
893 root 940 S vis -j ath0 eth0
894 root 940 S vis -j ath0 eth0
895 root 940 S vis -j ath0 eth0
```

Figura 4.10. Vista de batmand y vis en los procesos del Router Dragino.

Al ejecutar la instrucción **batmand –cd1** se obtienen los parámetros de todos los nodos a los cuales el router Dragino puede tener comunicación.

```
root@192.168.1.102:~$ batmand -cd1
MainIF/IP: ath0/10.130.1.103, UT: 0d 0h11m]
10.130.1.213    ( 27)      10.130.1.1 [      ath0]:      10.130.1.1 ( 27)
10.130.1.227    ( 85)      10.130.1.1 [      ath0]:      10.130.1.1 ( 85)
10.130.1.222    ( 94)      10.130.1.1 [      ath0]:      10.130.1.1 ( 94)
10.130.1.214    ( 34)      10.130.1.1 [      ath0]:      10.130.1.1 ( 34)
10.130.1.224    (117)      10.130.1.1 [      ath0]:      10.130.1.1 (117)
10.130.1.215    ( 22)      10.130.1.1 [      ath0]:      10.130.1.1 ( 22)
10.130.1.212    ( 16)      10.130.1.1 [      ath0]:      10.130.1.1 ( 16)
10.130.1.204    ( 21)      10.130.1.1 [      ath0]:      10.130.1.1 ( 21)
10.130.1.218    ( 72)      10.130.1.1 [      ath0]:      10.130.1.1 ( 72)
10.130.1.226    (126)      10.130.1.1 [      ath0]:      10.130.1.1 (126)
10.130.1.228    ( 37)      10.130.1.1 [      ath0]:      10.130.1.1 ( 37)
10.130.1.221    (132)      10.130.1.1 [      ath0]:      10.130.1.1 (132)
10.130.1.205    ( 34)      10.130.1.1 [      ath0]:      10.130.1.1 ( 34)
10.130.1.207    ( 16)      10.130.1.1 [      ath0]:      10.130.1.1 ( 16)
10.130.1.203    ( 38)      10.130.1.1 [      ath0]:      10.130.1.1 ( 38)
10.130.1.225    (115)      10.130.1.1 [      ath0]:      10.130.1.1 (115)
10.130.1.231    ( 29)      10.130.1.1 [      ath0]:      10.130.1.1 ( 29)
10.130.1.217    (142)      10.130.1.1 [      ath0]:      10.130.1.1 (142)
```

Figura 4.11. Ejecución de batmand –cd1.

La Figura 4.11 muestra como el router Dragino tiene comunicación con los nodos de la red de medidores, la capacidad de poder adherirse a la red como un nodo más, así poder obtener los datos del consumo de energía que muestra los diferentes medidores instalados en la red de medidores de la Universidad.

4.3. PRUEBAS DE COMUNICACIÓN ENTRE ROUTER DRAGINO Y RED DE MEDIDORES.

Las pruebas realizadas en la comunicación del router Dragino con la red de medidores, es hacer **ping** a las direcciones IP de los dispositivos, empezando con el NS2, por medio de la red cableada, como se muestra en la Figura 4.12.

```
root@dragino-1988ca:/# ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=64 time=8.774 ms
64 bytes from 192.168.1.100: seq=1 ttl=64 time=1.602 ms
64 bytes from 192.168.1.100: seq=2 ttl=64 time=1.876 ms
64 bytes from 192.168.1.100: seq=3 ttl=64 time=1.873 ms
64 bytes from 192.168.1.100: seq=4 ttl=64 time=1.609 ms
```

Figura 4.12. Ping a 192.168.1.100.

Se realiza la comunicación entre el router Dragino y un medidor vía WIFI, haciendo ping al medidor instalado en la facultad de Agronomía, de la UES.

```
root@dragino-1988ca:/# ping 10.130.1.217
PING 10.130.1.217 (10.130.1.217): 56 data bytes
64 bytes from 10.130.1.217: seq=2 ttl=63 time=14.495 ms
64 bytes from 10.130.1.217: seq=4 ttl=63 time=11.351 ms
64 bytes from 10.130.1.217: seq=5 ttl=63 time=27.427 ms
64 bytes from 10.130.1.217: seq=10 ttl=63 time=10.163 ms
64 bytes from 10.130.1.217: seq=11 ttl=63 time=8.596 ms
```

Figura 4.13. Ping a medidor de Agronomía.

Se realiza la comunicación entre el router Dragino y un medidor por medio de la interfaz “alias”, haciendo ping al medidor instalado en la facultad de Agronomía, de la UES.

```
root@dragino-1988ca:/# ping 10.30.217.1
PING 10.30.217.1 (10.30.217.1): 56 data bytes
64 bytes from 10.30.217.1: seq=0 ttl=63 time=14.629 ms
64 bytes from 10.30.217.1: seq=1 ttl=63 time=12.117 ms
64 bytes from 10.30.217.1: seq=3 ttl=63 time=6.266 ms
64 bytes from 10.30.217.1: seq=5 ttl=63 time=7.804 ms
```

Figura 4.14. Ping a medidor de Agronomía por interfaz “alias”.

Se puede dar como exitosa la comunicación entre el router Dragino y la red de medidores de la UES.

4.4. PRUEBA DE DISPOSITIVO DE ALMACENAMIENTO CON MEDIDOR DE LA FACULTAD DE MEDICINA.

Se configura la IP del código C, con la IP del medidor instalado en la Facultad de Medicina de la UES. La cual es 10.30.227.2.

Posteriormente se procede a realizar la compilación cruzada con la herramienta Buildroot, como lo muestra la Figura 4.15.

```
marvin_cornejo@ubuntu:~/Escritorio/tesis_marvin/Dragino$
‘/home/marvin_cornejo/opt/toolchains-mips/buildroot-2011.11/host/usr/bin/mips-
linux-gcc’ medidor_medicina.c –o m_medicina `pkg-config –libs –cflags libmodbus`
```

Figura 4.15. Compilación cruzada para medicina.c

Se copia el archivo creado por Buildroot al router Dragino.

```
Marvin_cornejo@ubuntu:~$ scp m_medicina root@192.1681.102:/
```

Figura 4.16. Copia desde pc a router Dragino.

Se configura crontab para que realice la ejecución del script write_uart.lua, encargado de la ejecución del archivo compilado y envió de los datos al puerto serial de la placa Arduino.

La Figura 4.17 muestra la configuración de crontab[20], se utilizan los cinco asteriscos, indicando que la ejecución del script Lua se realizará cada minuto, se puede configurar para obtener los datos del medidor cada cinco minutos como se muestra en la Figura 4.18.

```
root@dragino-1988ca:/# crontab –e
   * * * * *      /Tesis/lua write_uart.lua
```

Figura 4.17. Uso de crontab para ejecución de script write_uart.lua

```
root@dragino-1988ca:/# crontab –e
   5 * * * *      /Tesis/lua write_uart.lua
```

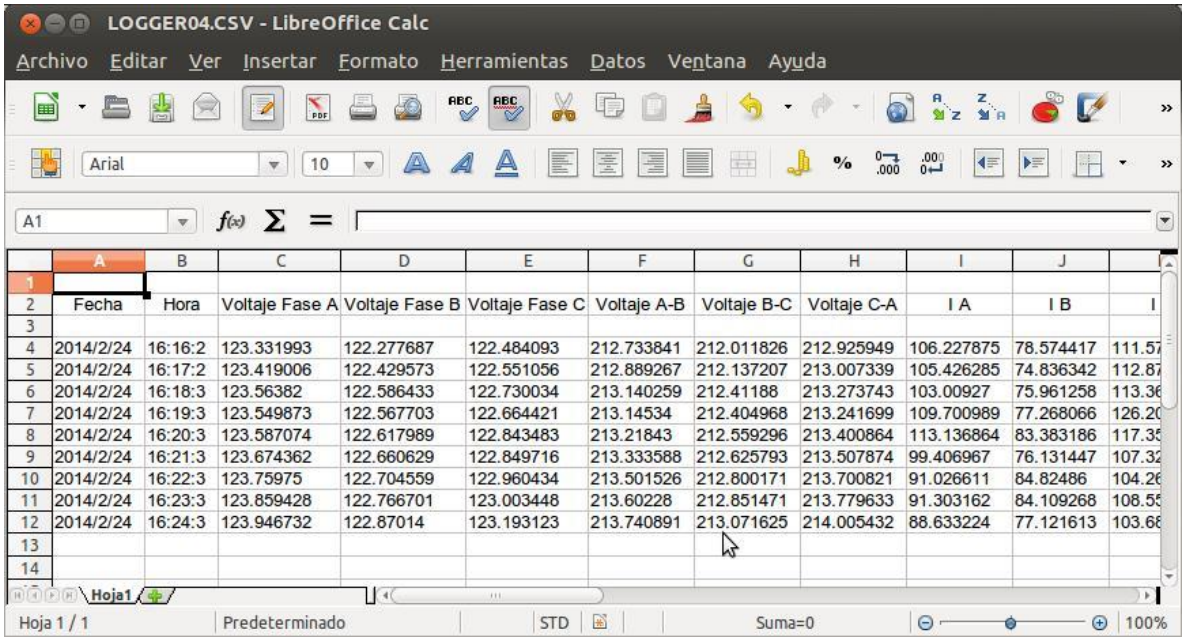
Figura 4.18. Ejecución de script Lua cada cinco minutos.

Una vez realizado los pasos anteriores se procede al reinicio del router Dragino, para que crontab empiece a ejecutar la orden que se le ha dado desde el arranque del sistema. Se vuelve a ingresar al router por medio de ssh.

La obtención de los datos obtenidos del medidor instalado en medicina se muestra en la Figura 4.19.

```
Marvin_cornejo@ubuntu:~$ ssh root@192.168.1.102
Password:
```

Figura 4.19. Ejecución de ssh para ingresar a router Dragino.



	A	B	C	D	E	F	G	H	I	J
1										
2	Fecha	Hora	Voltaje Fase A	Voltaje Fase B	Voltaje Fase C	Voltaje A-B	Voltaje B-C	Voltaje C-A	I A	I B
3										
4	2014/2/24	16:16:2	123.331993	122.277687	122.484093	212.733841	212.011826	212.925949	106.227875	78.574417
5	2014/2/24	16:17:2	123.419006	122.429573	122.551056	212.889267	212.137207	213.007339	105.426285	74.836342
6	2014/2/24	16:18:3	123.56382	122.586433	122.730034	213.140259	212.41188	213.273743	103.00927	75.961258
7	2014/2/24	16:19:3	123.549873	122.567703	122.664421	213.14534	212.404968	213.241699	109.700989	77.268066
8	2014/2/24	16:20:3	123.587074	122.617989	122.843483	213.21843	212.559296	213.400864	113.136864	83.383186
9	2014/2/24	16:21:3	123.674362	122.660629	122.849716	213.333588	212.625793	213.507874	99.406967	76.131447
10	2014/2/24	16:22:3	123.75975	122.704559	122.960434	213.501526	212.800171	213.700821	91.026611	84.82486
11	2014/2/24	16:23:3	123.859428	122.766701	123.003448	213.60228	212.851471	213.779633	91.303162	84.109268
12	2014/2/24	16:24:3	123.946732	122.87014	123.193123	213.740891	213.071625	214.005432	88.633224	77.121613
13										
14										

Figura 4.20. Datos almacenados en archivo Logger04.csv.

Se tiene que tomar en consideración la configuración de la hora en la que se adquieren los datos, se tiene un desfase de tiempo de aproximadamente 5 minutos, esto debido a que el reloj de tiempo real RTC, del módulo Datalogger se sincroniza con la señal que recibe de la placa Arduino, dicha placa debe estar conectada a la pc para poder reconfigurar la hora del RTC. Es por ello que antes de iniciar con las mediciones se tiene que conectar la placa Arduino con el módulo Datalogger a la pc, y luego dejar trabajando solo con el router Dragino. Este paso configura la hora de trabajo, pero al conectar la placa Arduino al router Dragino se pierde cierto tiempo y es ahí donde se da el error.

4.5. AUMENTO DE CAPACIDAD DE BUFFER EN PUERTO SERIAL DE ARDUINO.

Otro aspecto a tomar en cuenta es sobre la capacidad de buffer que tiene el puerto serial de la placa Arduino a la hora de recibir los datos, ya que el buffer asignado al puerto serial solo tiene la capacidad de recibir 128 bytes, uno a uno. Es por ello que se realiza una modificación al archivo HardwareSerial.cpp[21], que se encuentra en **/usr/share/arduino/hardware/arduino/cores/.**

Para realizar dichos cambios, se tiene que hacer una copia del directorio arduino, al que le llamamos arduino_256_buff. En el cual se realiza los cambios al archivo HardwareSerial.cpp.

En la línea que está situada en la parte superior del archivo HardwareSerial.cpp #define SERIAL_BUFFER_SIZE 64, se sustituye por la línea #define SERIAL_BUFFER_SIZE 256.

Realizado el cambio se procede a modificar el archivo boards.txt.

Realizado los cambios, Arduino cuenta con una nueva configuración de placa,

para utilizar el puerto serial con 256 bytes. Como se observa en la Figura 4.21.

```
uno.name = Arduino Uno
uno.upload.protocol = arduino
uno.upload.maximum_size = 32256
uno.upload.speed = 115200
uno.bootloader.low_fuses = 0xff
uno.bootloader.high_fuses = 0xDE
uno.bootloader.extended_fuses = 0x05
uno.bootloader.path = optiboot
uno.bootloader.file = optiboot_atmega328.hex
uno.bootloader.unlock_bits = 0x3F
uno.bootloader.lock_bits = 0x0F
uno.build.mcu = ATMEGA328P
uno.build.f_cpu = 16000000L
uno.build.core = arduino
uno.build.variant = estándar
##############################################################
uno256.name = Arduino Uno (256 Buffer Serial)
uno256.upload.protocol = arduino
uno256.upload.maximum_size = 32256
uno256.upload.speed = 115200
uno256.bootloader.low_fuses = 0xff
uno256.bootloader.high_fuses = 0xDE
uno256.bootloader.extended_fuses = 0x05
uno256.bootloader.path = optiboot
uno256.bootloader.file = optiboot_atmega328.hex
uno256.bootloader.unlock_bits = 0x3F
uno256.bootloader.lock_bits = 0x0F
uno256.build.mcu = ATMEGA328P
uno256.build.f_cpu = 16000000L
uno256.build.core = arduino_256_serialbuf
uno256.build.variant = estándar
```

Sección agregada para la utilización de una nueva tarjeta

Figura 4.21. Modificación de archivo boards.txt en Arduino.

Realizado los cambios ahora se tiene que seleccionar en el menú de Tools, el submenú Board donde se encontrara la configuración de la nueva tarjeta como Arduino uno (256 Serial Buffer).

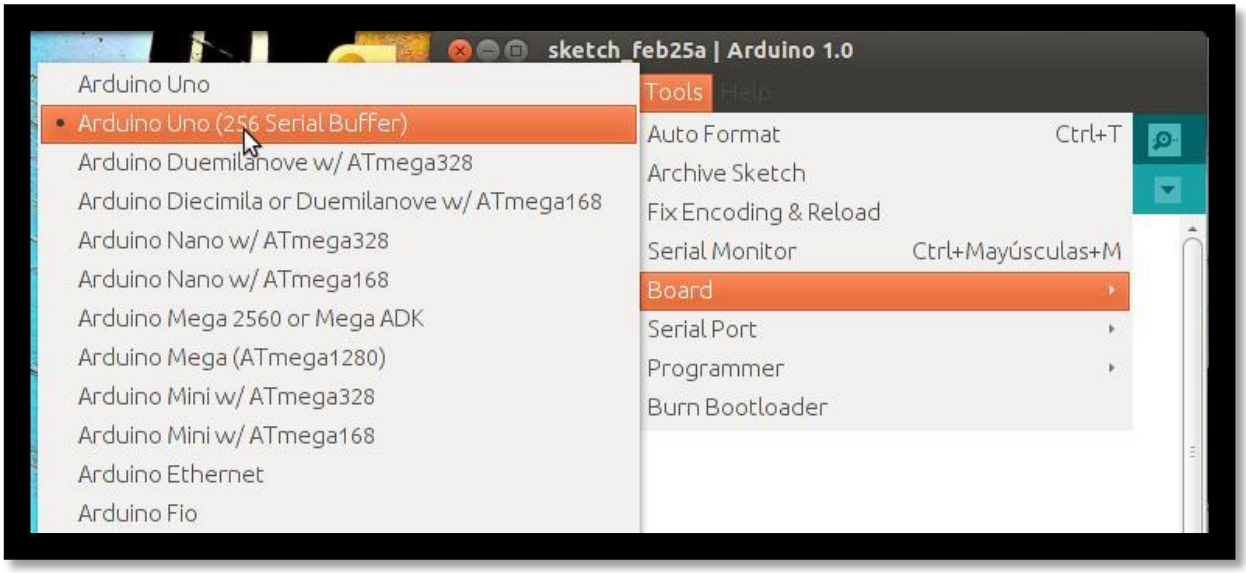


Figura 4.22. Modificación de board en Arduino.

Con las modificaciones anteriores, se puede tener hasta 24 datos recibidos de algún medidor de la red de la UES que esté conectado al dispositivo de almacenamiento.

Una vez realizado todas las configuraciones descritas en este capítulo la aplicación del router Dragino como dispositivo de almacenamiento masivo se da por exitosa, ya que se logró la conjunción de dos dispositivos como el router Dragino y la placa Arduino con su módulo Datalogger SD, para tener un respaldo de los datos generados por la red de medidores de la UES.

CONCLUSIONES

- Con el presente trabajo se muestra una de las aplicaciones de las cuales puede ser utilizado el router Dragino, como dispositivo de almacenamiento de datos, con la finalidad de tener un ahorro tanto en la construcción del dispositivo como en el consumo de energía que este genera.

- La implementación de la herramienta Buildroot para la realización de la compilación cruzada, el mayor logro del presente trabajo, ya que se puede ejecutar el protocolo Modbus escrito en lenguaje C en el router Dragino, sabiendo de las limitantes que este posee, con respecto de no contar con su propio compilador para dichos códigos. A la vez de la ejecución de dicho código por medio de un script en lenguaje de programación Lua, el cual es el qu tiene control del puerto UART del router Dragino.

- La comunicación de la placa Arduino con el router Dragino, por medio de sus puertos, serial para Arduino y UART para Dragino, da paso a la transferencia de datos entre ambos dispositivos.

- Con el módulo Datalogger SD, se tiene la capacidad de poder almacenar los datos obtenidos por el router Dragino en un script bajo el formato que se desee, es aquí donde se da la capacidad al dispositivo formado por el router Dragino y la placa Arduino la capacidad de almacenamiento de hasta 4GB.

ANEXO A.

A.1. ROUTER DRAGINO.

Figura A.1. Router Dragino.

Sistema Operativo OpenWrt incorporado.

OpenWrt es una distribución Linux fiable para los sistemas de embebidos. Es de código abierto, bajo buen desarrollo y mantenimiento. OpenWrt usa sistema de paquetes, muchos de gran utilidad. Los usuarios pueden instalar, eliminar estos paquetes para hacer el sistema de MS12 personalizado utilizando sistema operativo OpenWrt[21] y ofrece un sistema potente.

Diseño de Hardware abierto.
Los usuarios tienen la posibilidad de mejorar o realizar innovación en el hardware y lanzar un nuevo producto basado en sus requerimientos.

Modo de conexión a la red.

Se puede configurar en modo WiFi para adaptarse a diferentes aplicaciones/requisitos de red. Puede configurar el dispositivo así:

1. **Modo AP** como lo hace un router normal WiFi, por lo que los demás clientes WiFi se puede conectar a MS12 y compartir su conexión Internet.

2. **Modo AP Cliente** como lo hace PC normal para conectarse a su router WiFi existente para conectarse a Internet
3. **Modo Ad-hoc** y construir una red en malla

Diseño modular.

Lo más complicado y parte más difícil es la construcción de la placa base (motherboard). Los desarrolladores se centran en el diseño modular de MCU, conocida en inglés como daugther board. Esto ayudará a reducir el tiempo de desarrollo, el coste, la dificultad y el riesgo para el desarrollador para diseñar o probar un nuevo producto para otro tipo de aplicaciones distintas a la medición de energía eléctrica.

Características Técnicas del Router Dragino MS12.

A continuación a manera de introducción a las características del router, la Figura 6 muestra una clara perspectiva del interior del dispositivo, además se puede observar con total claridad la distribución de los principales componentes en el MS12.

Descripción general del sistema	**Especificaciones Inalámbricas**	***Interfaz de microcontrolador***
• CPU: Atheros AR2317, procesador de 180MHz, MIPS 4K, Flash de 8 MB, RAM de 16 MB • 1 x puerto Ethernet 10/100M • Distribución OpenWRT de Linux • Acceso completo vía SSH • Acceso remoto para control y actualización del MCU • Corriente continua de entrada: 9 ~ 15V.	• Estándar IEEE 802.11b / g • Banda de frecuencia: 2.4 ~ 2.462GHz • Antena externa • Potencia WiFi 20 dBm • Capacidad de funcionar en modo WiFi AP, cliente o Ad-hoc	• 12 puestos de terminales atornillabes • GPIO de AR2317, se puede simular como SPI • UART

Tabla A.1. Especificaciones técnicas de router Dragino.

A.1.1. Arquitectura del Dragino MS12.

En la imagen que se muestra a continuación (Figura A.2), se exhibe la arquitectura del router Dragino MS12, dentro de la cual se especifica a groso modo los principales componentes o partes que forman el dispositivo.

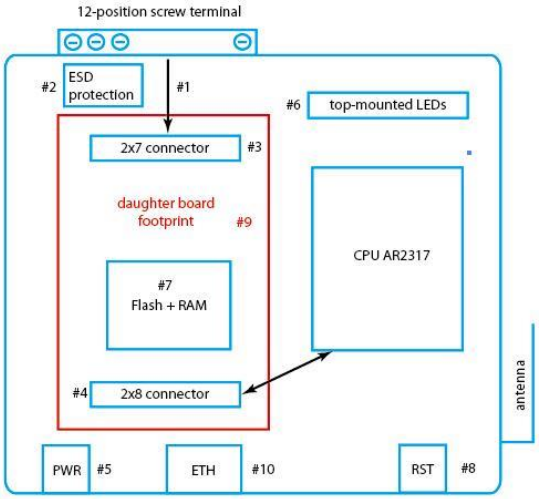


Figura A.2. Arquitectura interna del Dragino MS12.

Especificación de los principales componentes del Dragino MS12:

1: Puente entre el borne de tornillo de 12 posiciones y conector 2x7

2: Componentes de protección ESD

3: Conectores 2x7.

4: GPIO (SPI), UART, LED externa, COLD_RST_L, 3.3 V / 5 V

5: Puerto de alimentación.

6: LED's indicadores del funcionamiento (WiFi, Heartbeat, LAN, SYS, PWR).

7: 8 MB Flash + 16 MB RAM.

8: RST.

9: Espacio físico utilizado por de módulo Dragino u otro dispositivo.

#10: Puerto Ethernet.

A.2. INSTALACION DE FIRMWARE ROUTER DRAGINO.

Se procede a descargar el archivo para el flash del router, ap51-flash.

http://svn.dragino.com/tools/.

Figura A.3. Dirección de descarga de ap-51-flash.

Para la instalación del firmware se tiene acceso a la dirección.

http://www.dragino.com/downloads/index.php?dir=motherboards/ms12.

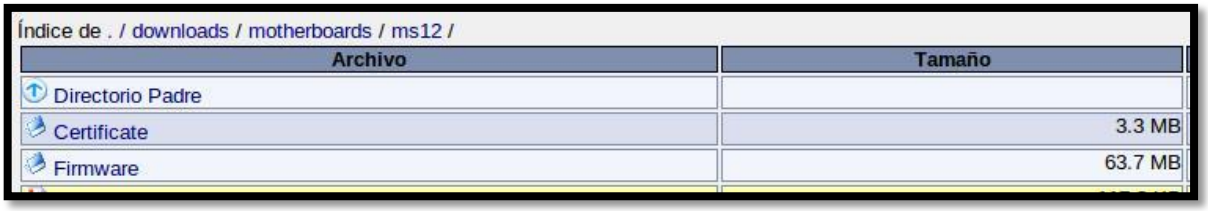


Figura A.4. Descarga de archivos para instalación.

En la opción firmware se encuentra las versiones que se pueden instalar en el router, en este caso se instaló la versión 2.1-beta3.

La versión 2.1-beta3 tiene la cualidad de añadir en modo trasparente rx a UART.

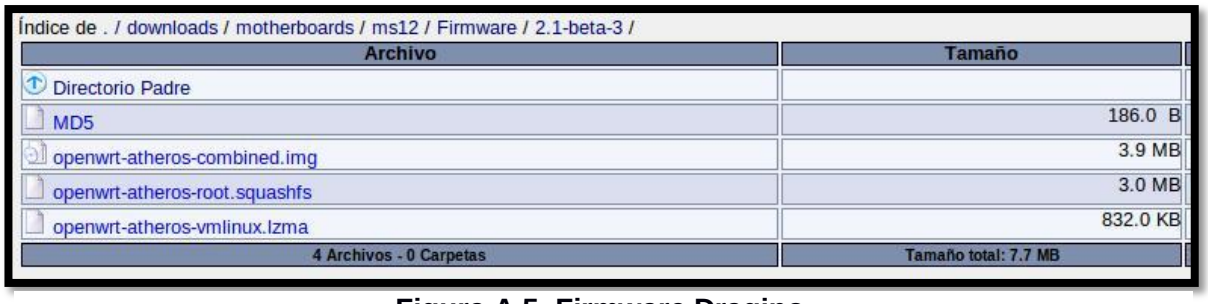


Figura A.5. Firmware Dragino.

A.3. PASOS PREVIOS A INSTALACION DE FIRMWARE.

Una vez descargados los archivos se colocan en un directorio para tener acceso al momento de la instalación.

Se siguen los siguientes pasos:

Paso 1: se accede desde consola al directorio donde se encuentran los archivos de instalación.

Figura A.6. Directorio con archivos de instalación.

.

Paso 2: desde una terminal, se inicia como súper usuario y se dan permisos de ejecución al archivo ap51-flash.

```
marvin_cornejo@ubuntu:~$ sudo su
[sudo] password for marvin_cornejo:
root@ubuntu:/home/marvin_cornejo# cd dragino/
root@ubuntu:/home/marvin_cornejo/dragino# chmod +x ap51-flash
```

Figura A.7. Permisos de ejecución de ap51-flash.

Paso 3: se deshabilita la conexión a WIFI de la pc.

Paso 4: se conecta el cable RJ-45 en el puerto Ethernet del Dragino y al de la computadora, el router no tiene que estar conectado a la fuente de poder. Como lo muestra la figura A.8.

Figura A.8. Conexión de Dragino con PC.

Paso 5: se configura la IP de nuestra computadora dentro del rango al que pertenece el router Dragino.

```
marvin_cornejo@ubuntu:~$ sudo ifconfig eth0 192.168.255.2/24 up
```

Figura A.9. Configuración de IP en PC.

A.4. INSTALACIONDE FIRMWARE DRAGINO.

Con permisos de súper usurario se procede a ejecutar la línea de comando que se muestra en la figura A.10.

```
marvin_cornejo@ubuntu:~$ ./ap51-flash openwrt-atheros-root.squashfs
openwrt-atheros-vmlinux.lzma
```

Figura A.10. Instalación de Dragino.

Una vez se ejecute la acción, Linux buscara al router por medio del puerto Ethernet, como lo muestra la figura A.11.

```
Device detection in progress . . . . .
```

Figura A.11. Detección de router en el proceso de instalación.

Al observar el mensaje anterior se procede a conectar el router Dragino a la fuente de poder para proceder a la instalación del firmware.

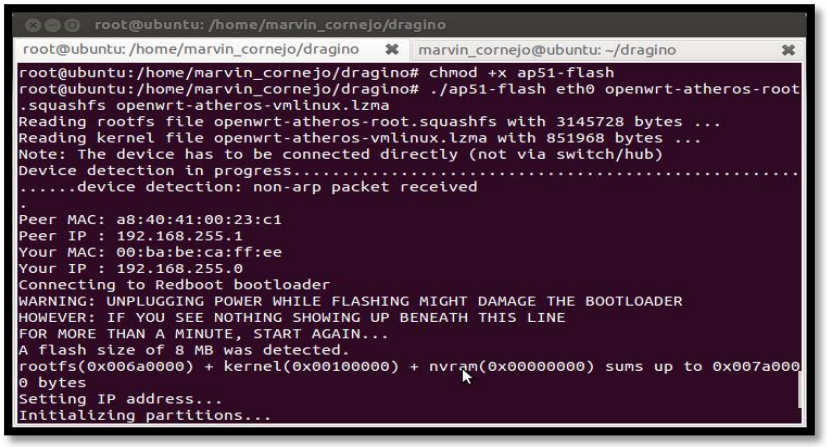


Figura A.12. Proceso de instalación de software.

Una vez se termine la instalación, el router se reiniciara, teniendo como IP de acceso la 192.168.255.1.

A.5. ACCESO A ROUTER DRAGINO.

Para tener acceso al router se tiene de varias formas, una por medio de su interfaz gráfica y la otra mediante ssh[22].

.

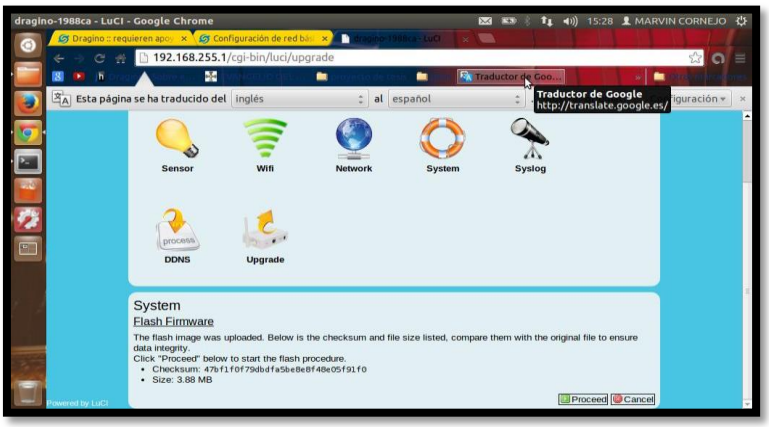


Figura A.13. Interfaz gráfica de Dragino.

Como se muestra en la Figura A.13. Se tiene acceso desde un navegador de internet, utilizando la dirección IP del router. Otra forma es mediante ssh, se puede visualizar los directorios que contiene el software del sistema, que para nuestro caso es OpenWrt backfire 10.03.1, como se observa en la siguiente figura.

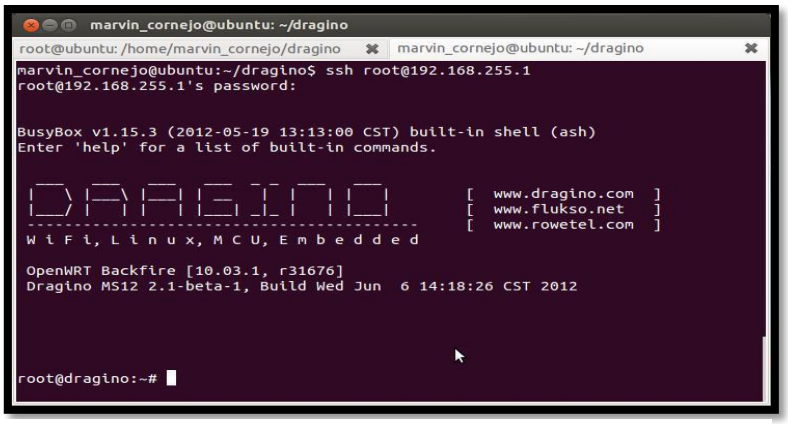


Figura A.14. Vista de acceso mediante ssh.

A.6. PLACA ARDUINO.

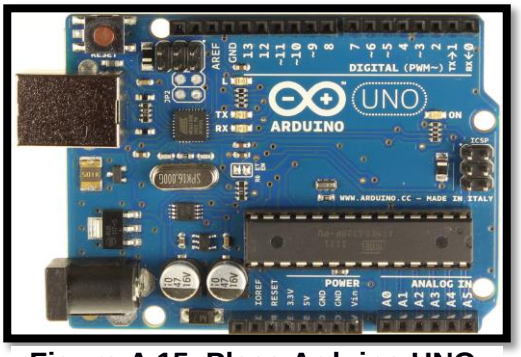


Figura A.15. Placa Arduino UNO.

Es un hardware libre, basado en una placa que cuenta con un microcontrolador y un entorno de desarrollo. El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son elAtmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (*boot loader*) que corre en la placa.

Tiene E/S análogas como digitales, así como E/S avanzadas, interrupciones y comunicación por puerto serial.

A.7. DATALOGGER SD.

Datalogger Shield en esencia es un módulo compatible con Arduino, adaptable a la placa Arduino, ya que con este se puede almacenar datos en una SD en tiempo real. Este dispositivo se obtiene en forma de kit para armar, además contiene un reloj en tiempo real (RTC).

Características:

- Soporta formatos en FAT16 o FAT32.

- Circuito adaptador de nivel de voltaje para evitar riesgos de dañar las SD card.
- Librerías para la implementación de códigos, tanto para la SD card como para el RTC[23].

Se observa que por medio de reloj en tiempo real (RTC), se puede almacenar los datos según la hora en la que ha sido adquirida la medición. Por medio del script en C, el cual es que ejecuta las órdenes para poder tener la comunicación entre el Arduino y el Datalogger SD. Este podrá almacenar según el formato de archivo que se desee, como por ejemplo formato txt, cvs, etc.

La comunicación entre el microcontrolador y la tarjeta SD es por medio del protocolo SPI, que tiene lugar en los pines digitales 11, 12 y 13 (en la mayoría de las placas Arduino) o 50, 51 y 52 (Arduino Mega).Además, otro pin debe ser utilizado para seleccionar la tarjeta SD. Este puede ser el hardware SS pin - pin 10 (en la mayoría de las placas Arduino) o pin 53 (en la Mega). U otro pin especificado en la llamada a SD.begin () es de tener en cuenta que incluso si no se utiliza SS pin, debe dejarse como una salida, sino la biblioteca SD no funcionará.

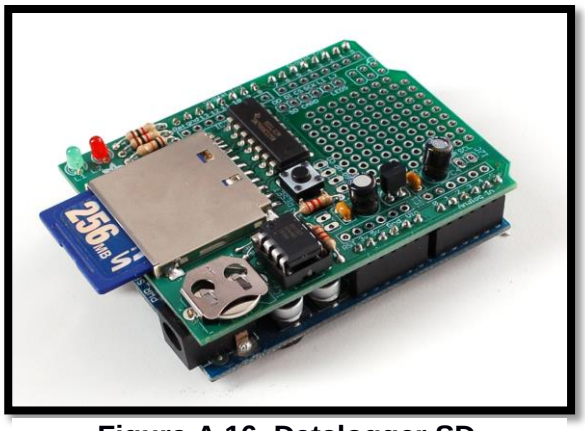


Figura A.16. Datalogger SD.

A.8. INSTALACION DE ARDUINO.

El software Arduino es libre, de código abierto disponible para Windows, mac OS y Linux.

En este caso se utilizó la versión 1.05 para Linux y Windows, disponible su descarga en la siguiente dirección: http://arduino.cc/en/Main/Software.

En Linux, también se puede tener acceso a la descarga de dicho software desde el gestor de paquetes o desde línea de comando de la siguiente forma:

```
Marvin_cornejo@ubuntu:~$ sudo apt-get install arduino
```

Figura A.17. Instalación de Aduino desde línea de comando.

A.9. EJECUCIÓN DE PROGRAMA ARDUINO.

Una vez instalado, se procede a ejecutar Arduino, el cual muestra la siguiente interfaz gráfica:

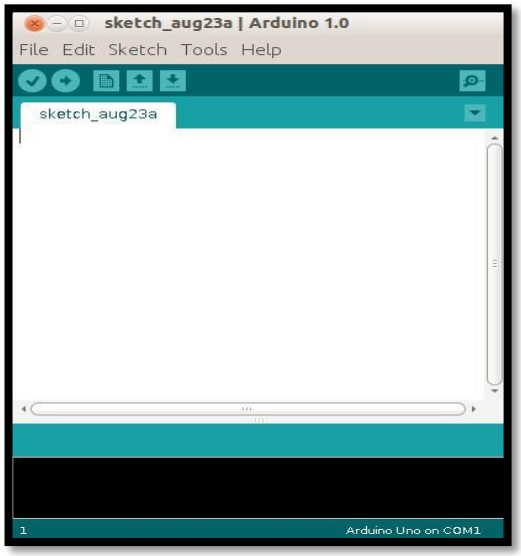


Figura A.18. Interfaz gráfica de Arduino IDE.

Al tener el Sketch abierto se procede a abrir algún archivo en el que se quiera trabajar de la siguiente forma:

- file>open>carpeta donde se encuentra archivo>medidor.ino

Como se muestra en la siguiente imagen:

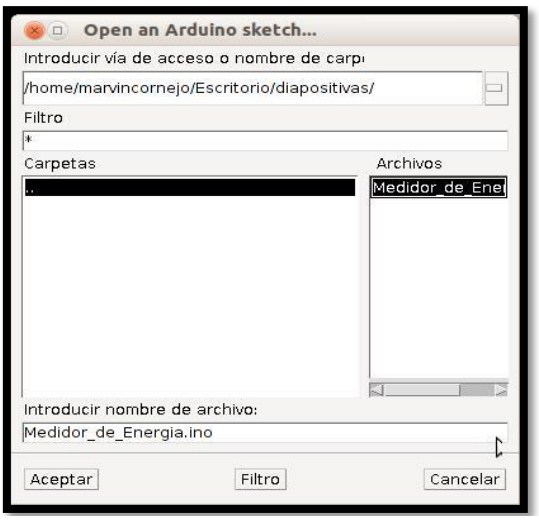


Figura A.19. Abrir archivo Arduino.

Una vez el código Arduino es ingresado al sketch, se procede a seleccionar el puerto en el cual se conectará la placa Arduino para poder “quemar” el código en el módulo como se muestra a continuación:

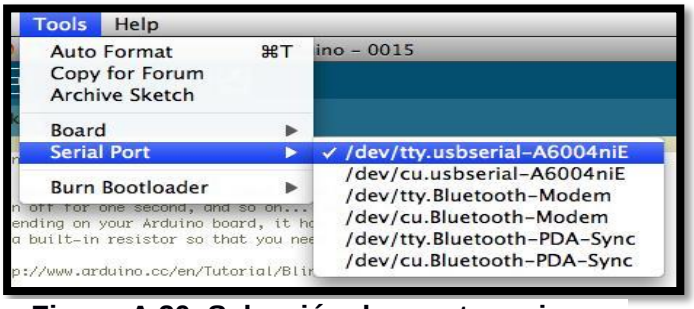


Figura A.20. Selección de puerto serie.

ANEXO B.

B.1. LIBRERIA LIBMODBUS.

Libmodbus es una biblioteca de software libre para enviar / recibir datos de acuerdo con el protocolo Modbus. Esta biblioteca está escrita en C y soporta RTU (de serie) y el TCP (Ethernet) de comunicaciones.
La librería Libmodbus que se utiliza en el proyecto es la 3.0.3 la cual se obtiene de la siguiente dirección.

https://github.com/downloads/stephane/libmodbus/libmodbus-3.0.3.tar.gz

B.2. FUNCIONES EN LIBMODBUS.

Por ser un conjunto de librerías, Libmodbus cuenta con varias funciones, las utilizadas en este trabajo son las siguientes:

- modbus_new tcp() : función que asigna e inicializa una estructura Modbus para su comunicación con un servidor Modbus[24].
- Modbus_connect(): función para la conexión con un servidor Modbus.[25]
- Modbus_free(): función que libera una estructura asignada a modbus_t.[26]
- Modbus_close(): función que debe cerrar una conexión establecida.[27]

Para un dispositivo que sea configurado como cliente, para el caso del router Dragino, se utiliza la siguiente funcion:

- modbus_read_registers(modbus_t *ctx, int addr, int nb, uint16_t *dest):función que lee los datos de nb, conteniendo registros para la dirección addr, el resultado se almacena en un registro *dest que debe ser de valor de una palabra(16bits).[28]

B.3. PROTOCOLO MODBUS.

Modbus es un protocolo de mensajería de capa de aplicación, situado en el nivel 7 del modelo OSI. Proporciona la comunicación cliente / servidor entre dispositivos conectados a diferentes tipos de buses o redes.
Modbus ha servido para que millones de dispositivos de automatización puedan comunicarse. Hoy en día, el apoyo a la estructura simple y elegante de Modbus sigue creciendo. La comunidad de Internet puede acceder a un sistema Modbus reservando el puerto 502 en la pila TCP / IP.

Modbus es un protocolo de solicitud / respuesta y ofrece servicios especificados por códigos de función.

B.4. INSTALACION DE LIBMODBUS EN PC.

Una vez descargado se procede a descomprimir el archivo. Se coloca el directorio en la carpeta personal y se procede a instalarlo como se muestra en la Figura B.1.

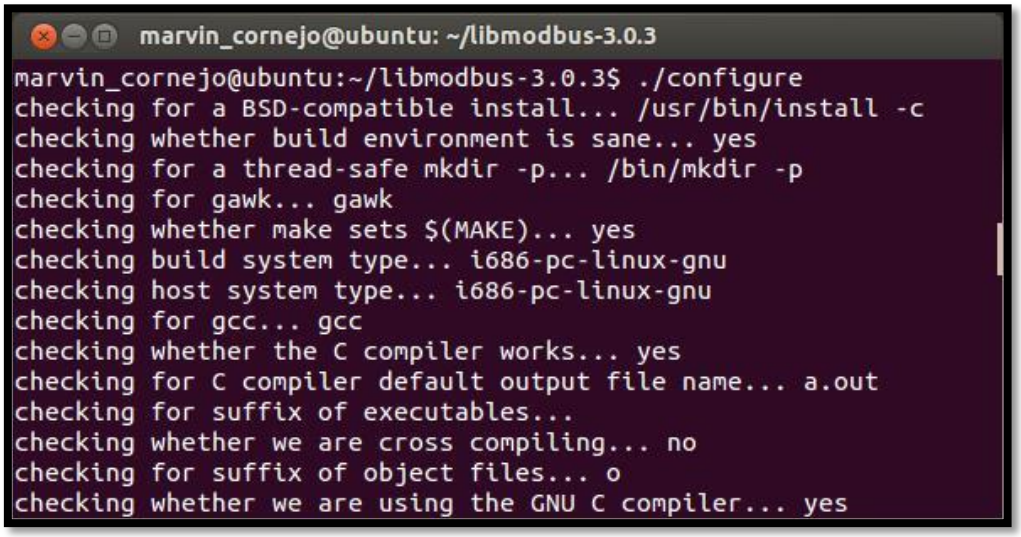


Figura B.1. Configuración de Libmodbus.

Se procede a ejecutar el comando make, tomando en cuenta que se debe hacer como usuario root. Como se observa en la figura B.2.

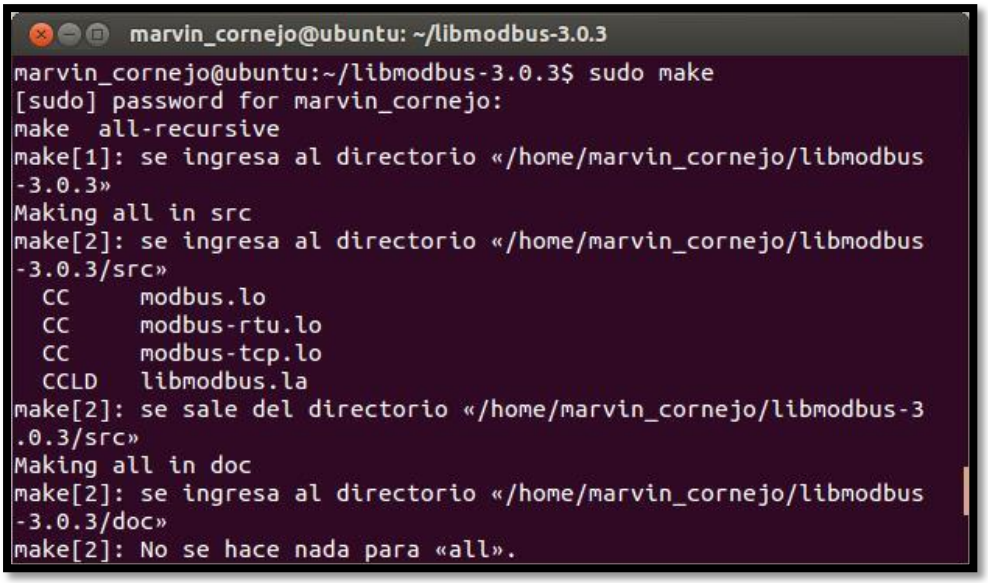


Figura B.2. Ejecución de make para Libmodbus.

Una vez concluido este paso se procede a ejecutar la siguiente línea siempre como usuario root, y ya se tiene instalada la librería Modbus en nuestro PC.

```
marvin_cornejo@ubuntu:~/libmodbus-3.0-3$ sudo make install
```

Figura B.3. Instalación de Libmodbus.

B.5. COMUNICACIÓN Y DISPOSITIVOS

Cada dispositivo destinado a comunicarse mediante Modbus se le da una dirección única. En las redes seriales sólo el nodo asignado como el Maestro puede iniciar un comando, pero en Ethernet, cualquier dispositivo puede enviar un comando Modbus, aunque por lo general sólo un dispositivo maestro lo hace.

Un comando Modbus contiene la dirección Modbus del dispositivo que está destinado. Sólo el dispositivo destinado actuará en el mando, a pesar de que otros dispositivos puedan recibirla.

Todos los comandos de Modbus contienen comprobación de la información, asegurando que una orden llegue en buen estado. Los comandos básicos Modbus pueden encargar a un RTU para cambiar un valor en uno de sus registros, control o leer un puerto de E / S, así como el dispositivo de comando para devolver uno o más valores que figuran en sus registros.

B.6. MODELO CLIENTE – SERVIDOR (MAESTRO/ESCLAVO).

El protocolo Modbus proporciona una comunicación cliente / servidor entre dispositivos conectados en una red Ethernet TCP / IP.[29]

Este modelo cliente / servidor se basa en cuatro tipos de mensajes:

- Solicitar MODBUS,
- Confirmación MODBUS,
- Indicación MODBUS,
- Respuesta MODBUS

Figura B.4. Modbus cliente-servidor.

B.7. ESTABLECIMIENTO DE LA CONEXIÓN.

Modbus debe proporcionar un socket de escucha en el puerto 502, que permite aceptar la nueva conexión y para intercambiar datos con otros dispositivos.
Cuando el servicio necesita intercambiar datos con un servidor remoto, debe abrir una nueva conexión de cliente con un puerto remoto 502 con el fin de intercambiar datos con este distante. El puerto local debe ser mayor que 1.024 y diferente para cada conexión de cliente.

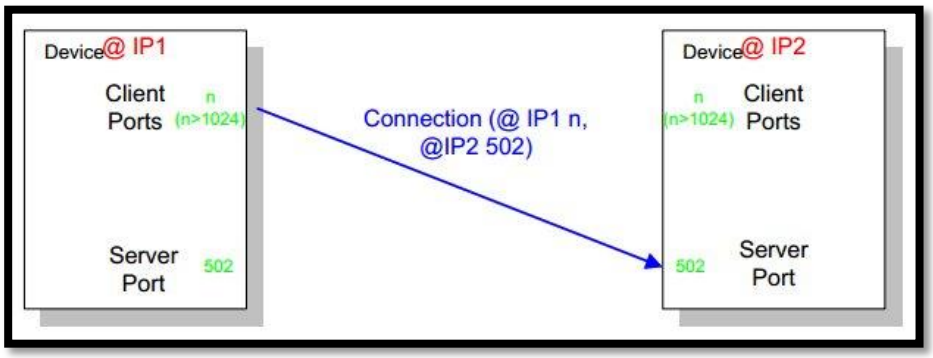


Figura B.5. Conexión de puertos.

Si el número de conexiones de cliente y el servidor es mayor que el número de conexiones autorizadas la conexión no utilizada más antigua se estará cerrando. El mecanismo de control de acceso puede ser activado para asegurarse si la

dirección IP del cliente remoto está autorizada. Si se rechaza no es una nueva conexión.

B.8. TRANSFERENCIA DE DATOS DE MODBUS

Una petición Modbus tiene que ser enviada en la conexión TCP ya abierta. La dirección IP del mando a distancia se utiliza para encontrar la conexión TCP. En caso de múltiples conexiones TCP abiertas con el mismo control remoto la conexión tiene que mantenerse abierta durante toda la comunicación.

Un cliente puede iniciar varias transacciones Modbus con el servidor sin tener que esperar el final de la anterior.
La comunicación que se tiene el módulo de almacenamiento de datos con la red de medidores es bajo el protocolo TCP/IP ya que en este se basa Modbus, es por ello de conocer ciertas características de este protocolo.
El protocolo TCP permite una comunicación fiable entre dos aplicaciones. De esta forma, las aplicaciones que lo utilicen no tienen que preocuparse de la integridad de la información: dan por hecho que todo lo que reciben es correcto.
Para que esta comunicación pueda ser posible es necesario abrir previamente una conexión. Esta conexión garantiza que los todos los datos lleguen correctamente de forma ordenada y sin duplicados. La unidad de datos del protocolo es el byte, de tal forma que la aplicación origen envía bytes y la aplicación destino recibe estos bytes. Como se muestra en la figura B.6:

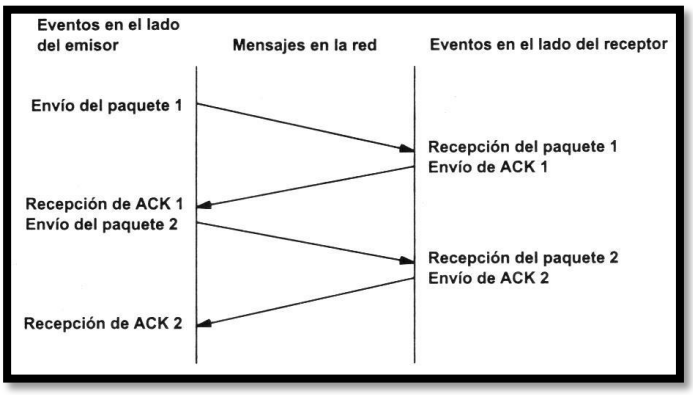


Figura B.6. Conexión TCP.

Sin embargo, cada byte no se envía inmediatamente después de ser generado por la aplicación, sino que se espera a que haya una cierta cantidad de bytes, se agrupan en un segmento y se envía el segmento completo. Para ello son necesarias unas memorias intermedias o buffers. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP.

El protocolo TCP envía un flujo de información no estructurado. Esto significa que los datos no tienen ningún formato, son únicamente los bytes que una aplicación envía a otra. Ambas aplicaciones deberán ponerse de acuerdo para comprender la información que se están enviando.

B.9. UTILIZACION DE WIRESHARK PARA TRAFICO DE DATOS.

Wireshark[30] se utiliza para identificar y analizar el tráfico en un momento determinado, es decir puede capturar todo el tráfico que circula por la red. Con este fin se utilizó en el proyecto, para conocer el comportamiento de los paquetes enviados por el medidor de energía hacia el router Dragino.

Algunas de las características de Wireshark son las siguientes:

- Permite obtener detalladamente la información del protocolo utilizado en el paquete capturado.
- Cuenta con la capacidad de importar/exportar los paquetes capturados desde/hacia otros programas.
- Filtra los paquetes que cumplan con un criterio definido previamente.
- Realiza la búsqueda de los paquetes que cumplan con un criterio definido previamente.
- Permite obtener estadísticas.

En el proyecto se utiliza el protocolo Modbus, en el cual se debe observar las siguientes características que muestra Wireshark:

ACK: Si este indicador está fijado en 1, el paquete es un acuse de recibo.

RST: Si este indicador está fijado en 1, se restablece la conexión.

SYN: El indicador SYN indica un pedido para establecer una conexión.

FIN: Si este indicador está fijado en 1, se interrumpe la conexión.

Como se puede observar en la Figura B.7.

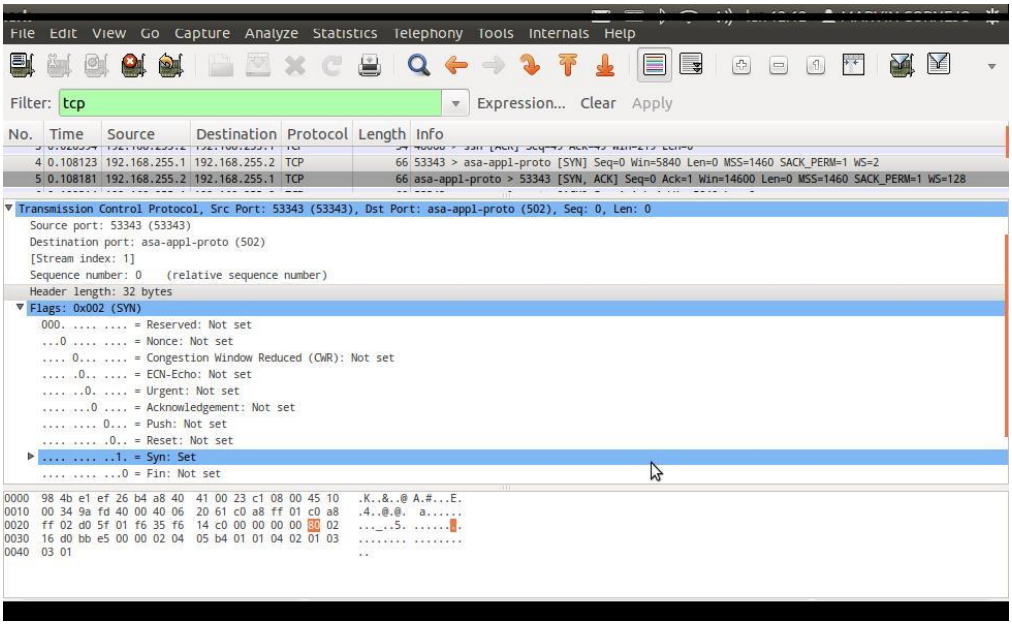


Figura B.7. Protocolo de transmisión SYN en Wirwshark.

Realizada la petición de conexión por la fuente, se espera la respuesta de parte del destinatario como se muestra en la Figura B.8.

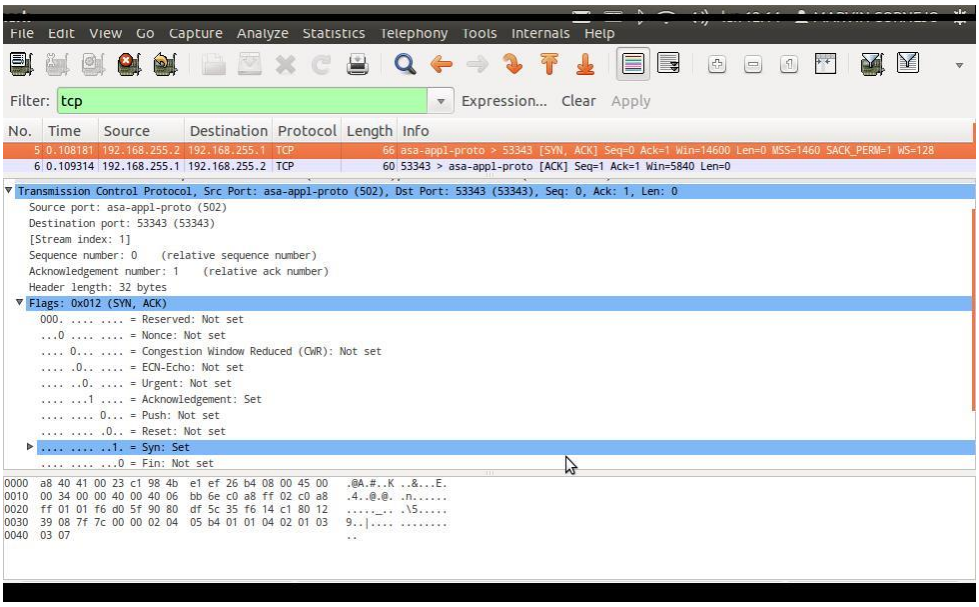


Figura B.8. Protocolo de transmisión SYN, ASK en Wirwshark.

Completada la transmisión de los paquetes, se procede al cierre de la conexión, como se muestra en la Figura B.9.

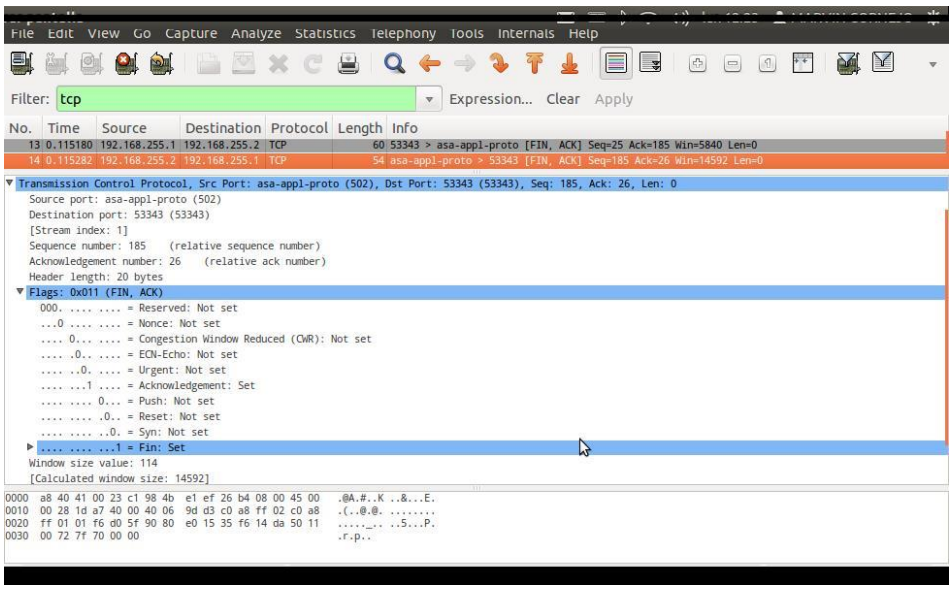


Figura B.9. Protocolo de transmisión FIN en Wirwshark.

B.9. INSTALACIÓN DE LUA Y LUA SOCKET EN PC.

Para la instalación de Lua, es necesario descargarlo de la siguiente dirección: http://www.lua.org/download.html. Es de mencionar que está la versión Lua 5.2, pero no es compatible para Lua socket.

Una vez realizado este paso se procede a la instalación, si es en Windows se ejecuta el archivo lua.5.1.exe, en Linux se puede hacer desde una terminal ejecutando la siguiente línea:

```
marvin_cornejo@ubuntu~$ sudo apt-get install lua5.1
```

Figura B.10. Instalación de lua desde terminal.

También al descomprimir e instalar el archivo lua 5.1.tar.gz de la siguiente forma:

```
marvin_cornejo@ubuntu:~$ curl –R –O http://www.lua.org/ftp/lua-
5.1.tar.gz
marvin_cornejo@ubuntu:~$ tar zxf lua-5.1.tar.gz
marvin_cornejo@ubuntu:~$ cd lua-5.1
marvin_cornejo@ubuntu:~$ make Linux test
```

Figura B.11. Instalación de lua5.1.tar.gz.

Se descarga la biblioteca Lua socket[31] desde la siguiente direccion:http://w3.impa.br/~diego/software/luasocket/home.html#download, se procede a la instalación como se muestra en la siguiente Figura B.12:

```
marvin_cornejo@ubuntu:~$ sudo apt-get install liblua5.2-socket2
[sudo] password for marvin_cornejo:
Creando arbol de dependencias
Leyendo la informacion de estado . . . hecho
Marvin_cornejo@ubuntu:~$
```

Figura B.12. Instalación de lua socket.

B.10. INSTALACIÓN DE LUA SOCKET EN ROUTER DRAGINO.

Para la instalación de la biblioteca Lua socket para el router Dragino, cuenta con el sistema operativo OpenWrt backfire 10.03.1 atheros; con esta especificación se procede a la búsqueda del paquete luasocket para este sistema, se descarga de la siguiente dirección: http://downloads.openwrt.org/backfire/10.03.1/atheros/packages.

Una vez realizado este paso se procede al envío del paquete luasocket_2.0.2-3_atheros.ipk a través de la siguiente línea de comando desde una terminal

```
Marvin_cornejo@ubuntu:~$ scp luasocket_2.0.2-3_atheros.ipk root@192.168.255.1:/
```

Figura B.13. Envío de paquete de PC a router con scp.

Realizado este paso se procede a instalar el paquete ipk, como se muestra en la Figura B.14:

```
root@192.168.255.1:~$ opkg install luasocket_2.0.2-3_atheros.ipk
```

Figura B.14. Instalación de luasocket en router Dragino.

El router Dragino cuenta con la capacidad de poder ejecutar los códigos escritos en lenguaje Lua que requieran la librería socket.

ANEXO C.

C.1. MEDIDOR SHARK.

Figura C.1. Medidor shark 200.

Electro Industries introduce un nuevo estándar en medición de montaje en panel. Shark es un dispositivo de medición de potencia y energía compacto que proporciona medición, clase de facturación combinado con funcionalidades como almacenamiento histórico de datos, comunicación de I/O que solo se encontraban tradicionalmente en sistemas de alto desempeño y costo[32].

Este producto está diseñado para incorporar características avanzadas en una cubierta pequeña, para aplicaciones eléctricas costo efectivas y en gran escala.

C.2. FUNCIONES BASICAS.

1. Medidor multifunción; se miden todos los parámetros de una red trifásica: V, I, cosφ, f, %THD, y potencia y energía.
2. Pulso KYZ para la certificación de la precisión; el equipo tiene típicamente una precisión del 0,1%, esto es verificable con la salida de pulso KYZ.
3. Puerto RS 485; protocolo Modbus y DNP 3.0
4. Barógrafo – Escala en % para tener una idea inmediata de la medida.

5. Programación simple e intuitiva.
6. Puerto óptico – Puesta en marcha y configuración posible desde una PDA o un portátil.
7. Minimización de opciones – Entradas con auto rrango y alimentación AC/DC.
8. Dos posibilidades para la entrada de corriente: sin conexión: el cable se pasa a través del equipo, con conexión: la forma típica Superior Current Inputs – Wire direct CT inputs or use Current "Gills".
9. Hueco necesario mínimo - es necesario únicamente un hueco de 92×92 mm en un panel.
10. Elevada relación calidad precio - es el equipo superior de su clase.
11. Tecnología V-Switch: un vez instalado y bien desde la red o el puerto óptico pueden aumentarse las funciones de medida:
    - V2 – Capacidad de registro de 2 MB.
    - V3 – V2 + Análisis de armónicos.
    - V4 – V3 + Alarmas y funciones de control.

C.3. CARACTERISTICAS TECNICAS.

Precisión - 0.1% ó superior para energía. Cumple ANSI C-12.20, IEC 687 Clase 0.2%

Comunicación – Protocolo Modbus 57.6 kb (RS 485 y puerto IRDA).

Frecuencia de muestreo – 400+ Samples per Cycle / 24 Bit A/D Resolución.

Protección entrada de corriente – 20x límite del rango por 10 segundos.

Tensión – 0 ~ 720 V L-L (precisión de 0,1% para 69~420 V L-N).

Corriente – 0 ~ 10 A (la precisión se garantiza para 0,15~5A).

Rango de frecuencias – 45 ~ 65 Hz.

C.4. MAPA MODBUS EN SHARK.

El mapa de Modbus para el medidor Shark proporciona detalles e información acerca de las posibles lecturas del medidor y su programación. El medidor Shark se puede programar con los botones en la carátula del medidor, o mediante el uso de software. Para una visión general de programación.

El Mapa de registro Modbus del medidor Shark incluye las siguientes secciones:

Sección de Datos Fijos, registros del 1 al 47, los detalles de información fija del medidor.

Sección datos del Medidor, Registros del 1000 al 12031, detalles de las lecturas del medidor, incluyendo lecturas Primaria, Bloque de Energía, Demanda de Bloque, Bloque Ángulo de Fase, Bloque de Estado, Bloque THD, Mínimos y Máximos en Regular y Bloques de Estampado de Tiempo, Bloques Opción de Tarjeta y Acumuladores. Modo de funcionamiento.

Sección Comandos, Registros del 20000 al 26011, detalles del Medidor, Bloque de restablecimiento, Bloque de programación, Otro bloque de comandos y Cifrado en bloque.

Sección de Ajustes programables, Registros del 30000 al 33575, todos los detalles de los ajustes se pueden programar para configurar su medidor. Sección Lecturas Secundaria.

Sección Lecturas Secundaria, Registros del 40001 al 40100, detalles del medidor, Lecturas secundarias.

Sección Recuperación de Registros, Registros del 49997 al 51095, los detalles de recuperación de registros.

C.5. FORMATO DE DATOS.

ASCII: Caracteres ASCII empaquetados 2 por registrarse en alto, bajo orden y Sin ningún carácter de terminación.

SINT16/UINT16: 16-bits con signo / sin signo entero.

SINT32/UINT32: 16-bits con signo / sin signo entero

FLOAT: 32-bit de punto flotante IEEE número que atraviesa 2 registros / sin Signo entero. El registro más bajo su dirección es la media de orden Superior (es decir, contiene el exponente).

C.6. MAPA MODBUS.

Modbus Address		Description (Note 1)	Format	Range (Note 6)	Unit or Resolution	Comments	# Reg
Hex	**Decimal**						
			Fixed Data Section				
Identification Block						**read-only**	
0000 - 0007	1 – 8	Meter Name	ASCII	16 char	None	t= transducer model (1=yes, 0=no).	8
0008 - 000F	9 – 16	Meter Serial Number	ASCII	16 char	None	S= submeter model(1 =yes,0=no).	8
0010 - 0010	17 - 17	Meter Type	UINT16	bit mapped	------st-----vvv	Vvv= V –switch: V1 = standard 200. V2 = V1 plus logging. V3 = V2 plus THD. V4 = V3 plus relays. V5 = V4 plus waveform capture up to 64 samples/cy cle and 3 Meg. V6 = V4 plus waveform capture up to 512 samples/cycle and 4 Meg	1
0011 – 0012	18 -19	Firmware Version	ASCII	4 char	none		2
0013 – 0013	20 - 20	Map Version	UINT16	0 to 65535	none		1
0014 - 0014	21 -21	Meter Configuration	UINT16	Bit –mapped	-----ccc --ttttt	ccc = CT denominator (1 or 5). Ffff ff = calibration frequency (50 or 60)	1
0015 – 0015	22 - 22	ASIC Version	UINT16	0-65535	none		1
0016 – 0017	23 - 24	Boot Firmware Version	ASCII	4 char	none		2
0018 - 0018	25 - 25	Option Sict 1 Usage	UINT16	bit -mapped	same as register 10000 (0x270F)		1
0019 - 0019	26 - 26	Option Slot 2 Usage	UINT16	bit -mapped	Same as register 11000 (0x2AF7)		1
001A – 001D	27 -30	Meter Type Name	ASCII	8 char	none		4
001E - 0026	31 – 39	Reserved				Reserved	9
0027 – 002E	40 – 47	Reserved				Reserved	8
002F - 0115	48 - 278	Reserved				Reserved	231
0116 - 0130	279 – 305	Integer Readings Block occupies these registers, see below					
0131 -01F3	306 – 500	Reserved				Reserved	194
01F4 - 0203	501 - 516	Reserved				Reserved	16

Tabla C.1. Información sobre el medidor.

Modbus Address		Description (Note 1)	Format	Range (Note 6)	Unit or Resolution	Comments	# Reg
Hex	Decimal						
Meter Data Section (Note 2)							
Readings Block (Integer Values)						**read-only**	
0116 - 0116	279 - 279	Volts A-N	UINT16	0 to 9999	volts		1
0117 - 0117	280 - 280	Volts B-N	UINT16	0 to 9999	volts		1
0118 - 0118	281 - 281	Volts C-N	UINT16	0 to 9999	volts		1
0119 - 0119	282 - 282	Volts A-B	UINT16	0 to 9999	volts		1
011A – 011A	283 - 283	Volts B-C	UINT16	0 to 9999	volts		1
011B – 011B	284 - 284	Volts C-A	UINT16	0 to 9999	volts		1
011C – 011C	285 - 285	Amps A	UINT16	0 to 9999	amps		1
011D – 011D	286 - 286	Amps B	UINT16	0 to 9999	amps		1
011E – 011E	287 - 287	Amps C	UINT16	0 to 9999	amps	1. Use the settings from Programmable settings for scale and decimal point location. (see User Settings Flags)	1
011F – 011F	288 - 288	Neutral Current	UINT16	-9999 to +9999	amps		1
0120 - 0120	289 - 289	Watts, 3Ph total	SINT16	-9999 to +9999	watts		1
0121 - 0121	290 - 290	VARs, 3Ph total	SINT16	-9999 to +9999	VARs	2. Per phase power and PF have values only for WYE hookup and will be zero for all other hookups.	1
0122 - 0122	291 - 291	Vas, 3-Ph total	UINT16	0 to +9999	VAs		1
0123 - 0123	292 - 292	Power Factor, 3Ph total	SINT16	-1000 to +1000	none		1
0124 - 0124	293 - 293	Frequency	UINT16	0 to 9999	Hz	3. If the reading is 10000 that means that the value is out of range. Please adjust the programmable settings in that case. The display will also show '…..' in case of over range.	1
0125 - 0125	294 - 294	Watts, Phase A	SINT16	-9999 M to +9999	watts		1
0126 - 0126	295 - 295	Watts, Phase B	SINT16	-9999 M to +9999	watts		1
0127 - 0127	296 - 296	Watts, Phase C	SINT16	-9999 M to +9999	watts		1
0128 - 0128	297 - 297	VARs, Phase A	SINT16	-9999 M to +9999 M	VARs		1
0129 – 0129	298 - 298	VARs, Phase B	SINT16	-9999 M to +9999 M	VARs		1
012A - 012A	299 - 299	VARs, Phase C	SINT16	-9999 M to +9999 M	VARs		1
012B - 012B	300 - 300	Vas, Phase A	UINT16	0 to +9999	VAs		1
012C - 012C	301 - 301	Vas, Phase B	UINT16	0 to +9999	VAs		1
012D - 012D	302 - 302	Vas, Phase C	UINT16	0 to +9999	VAs		1
012E - 012E	303 - 303	Power Factor, Phase A	SINT16	-1000 to +1000	none		1
012F - 012F	304 - 304	Power Factor, Phase B	SINT16	-1000 to +1000	none		1
0130 – 0130	305 - 305	Power Factor, Phase C	SINT16	-1000 to +1000	none		1
						Block Size:	**27**

Tabla C.2. Sección de datos del medidor.

Primary Readings Block						**Read -only**	
03E7 – 03E8	1000 – 1001	Volts A-N	FLOAT	0 to 9999 M	volts		2
03E9 – 03EA	1002 – 1003	Volts B-N	FLOAT	0 to 9999 M	volts		2
03EB – 03EC	1004 – 1005	Volts C-N	FLOAT	0 to 9999 M	volts		2
03ED – 03EE	1006 – 1007	Volts A-B	FLOAT	0 to 9999 M	volts		2
03EF – 03F0	1008 – 1009	Volts B-C	FLOAT	0 to 9999 M	volts		2
03F1 – 03F2	1010 – 1011	Volts C-A	FLOAT	0 to 9999 M	volts		2
03F3 – 03F4	1012 – 1013	Amps A	FLOAT	0 to 9999 M	amps		2
03F5 – 03F6	1014 – 1015	Amps B	FLOAT	0 to 9999 M	amps		2
03F7 – 03F8	1016 – 1017	Amps C	FLOAT	0 to 9999 M	amps		2
03F9 – 03FA	1018 – 1019	Watts, 3-Ph total	FLOAT	-9999 M to +9999 M	watts		2
03FB – 03FC	1020 – 1021	VARs, 3-Ph total	FLOAT	-9999 M to +9999 M	VARs		2
03FD – 03FE	1022 – 1023	Vas, 3-Ph total	FLOAT	-9999 M to +9999 M	VAs		2
03FF – 0400	1024 – 1025	Power Factor, 3Ph total	FLOAT	-1.00 to +1.00	none		2
0401 – 0402	1026 – 1027	Frecuency	FLOAT	0 to 65.00	Hz		2
0403 – 0404	1028 – 1029	Neutral Current	FLOAT	0 to 9999 M	amps		2
0405 – 0406	1030 – 1031	Watts, Phase A	FLOAT	-9999 M to +9999 M	watts	Per phase power and PF have values only for WYE hookup and will be zero for all other hookups.	2
0407 – 0408	1032 – 1033	Watts, Phase B	FLOAT	-9999 M to +9999 M	watts		2
0409 – 040A	1034 – 1035	Watts, Phase C	FLOAT	-9999 M to +9999 M	watts		2
040B – 040C	1036 – 1037	VARs, Phase A	FLOAT	-9999 M to +9999 M	VARs		2
040D – 040E	1038 – 1039	VARs, Phase B	FLOAT	-9999 M to +9999 M	VARs		2
040F – 0410	1040 – 1041	VARs, Phase C	FLOAT	-9999 M to +9999 M	VARs		2
0411 – 0412	1042 – 1043	Vas, Phase A	FLOAT	-9999 M to +9999 M	VAs		2
0413 – 0414	1044 – 1045	Vas, Phase B	FLOAT	-9999 M to +9999 M	VAs		2
0415 – 0416	1046 – 1047	Vas, Phase C	FLOAT	-9999 M to +9999 M	VAs		2
0417 – 0418	1048 – 1049	Power Factor, Phase A	FLOAT	-1.00 to +1.00	none		2
0419 – 041A	1050 – 1051	Power Factor, Phase B	FLOAT	-1.00 to +1.00	none		2
041B – 041C	1052 – 1053	Power Factor, Phase C	FLOAT	-1.00 to +1.00	none		2
041D – 041E	1054 – 1055	Symmetrical Component Magnitude, 0 Seq	FLOAT	0 to 9999 M	volts	Voltage unbalance per IEC 100-4.30 Values apply only to WYE hookup and will be zero for all other hookups.	2
041F – 0420	1056 – 1057	Symmetrical Component Magnitude, + Seq	FLOAT	0 to 9999 M	volts		2
0421 – 0422	1058 – 1059	Symmetrical Component Magnitude, - Seq	FLOAT	0 to 9999 M	volts		2
0423 – 0423	1060 – 1060	Symmetrical Component Phase, 0 Seq	UINT16	-1800 to +1800	0.1 degree		1
0424 – 0424	1061 – 1061	Symmetrical Component Phase, + Seq	UINT16	-1800 to +1800	0.1 degree		1
0425 – 0425	1062 – 1062	Symmetrical Component Phase, - Seq	UINT16	-1800 to +1800	0.1 degree		1
0426 – 0426	1063 – 1063	Unbalance, 0 sequence component	UINT16	0 to 65535	0.01%		1
0427 – 0427	1064 – 1064	Unbalance, -sequence component	UINT16	0 to 65535	0.01%		1
0428 - 0428	1065 - 1065	Current Unbalance	UINT16	0 to 20000	0.01%		1
						Block Size:	**66**

Tabla C.3. Sección de datos del medidor.

En la tabla C.1 muestra las direcciones de memoria para tener información acerca del medidor, como por ejemplo el nombre del medidor, su número de identificación, la versión del firmware, etc.

En la sección de datos podemos observar que mediante las direcciones que muestran las tablas C.2. Y C.3. Se tiene acceso a los datos obtenidos de los parámetros como voltaje en las líneas, sus respectivas corrientes, así como la energía que se consume.

ANEXO D.

D.1 INSTALACION DE SIMULADOR DIAGSLAVE.

El simulador Diagslave[33], es una herramienta en la cual se ejecuta el protocolo Modbus en modo esclavo, muy útil para sistemas de pruebas de dicho protocolo.
Es distribución libre, por lo que se encuentra disponible para los siguientes sistemas operativos y arquitecturas:

- Windows
- Linux (x86 32 bits)
- QNX neutrino 6 (x86)

Se utilizó dicha herramienta en Linux para realizar las pruebas del protocolo Modbus en el router Dragino, para ese fin se realizaron los siguientes pasos a seguir:

Paso 1: Descargar el simulador Diagslave desde la siguiente dirección:

http://www.modbusdriver.com/diagslave.html.

Paso 2: se descomprime el archivo diagslave.2.12.zip

Paso 3: se procede a posicionarse en la carpeta Linux, donde se encuentra un archivo ejecutable llamado Diagslave y se ejecuta la siguiente línea de comando:

```
Marvin_cornejo@ubuntu:~$ sudo ./diagslave
```

Figura D.1. Ejecución de diagslave en Linux.

Al ejecutarse la instrucción mostrada en la Figura D.1, se tiene que realizar con permisos de súper usuario para tener control del puerto Ethernet.

Se muestra el comportamiento de dicha herramienta en la misma Terminal en la Figura D.2.

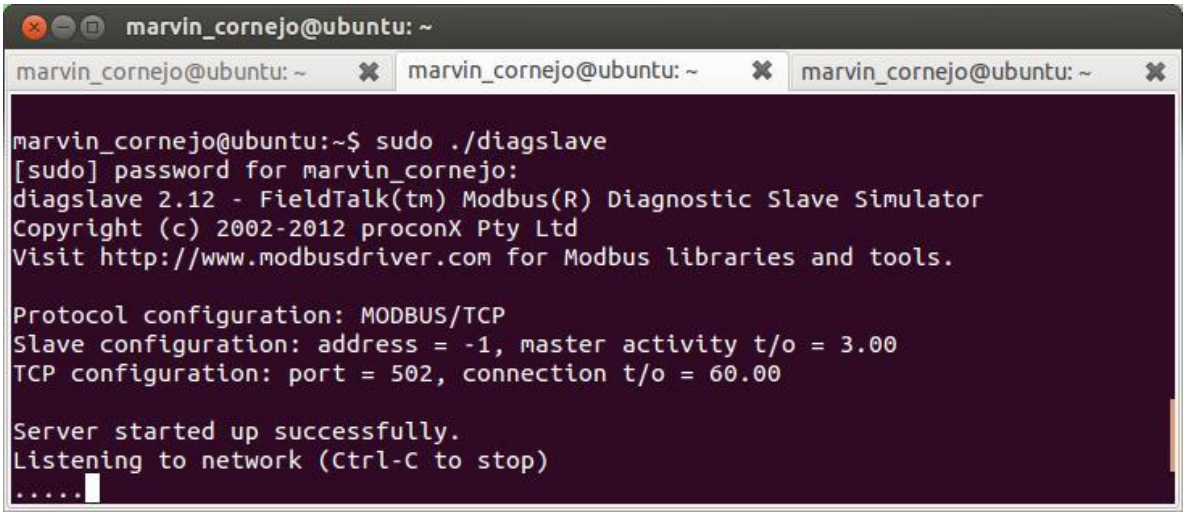


Figura D.2. Diagslave desde una Terminal en Linux.

Una vez realizado los pasos anteriores, el simulador Diagslave está listo para ser utilizado con el router Dragino.

D.2. PRUEBAS REALIZADAS CON EL SIMULADOR DIAGSLAVE Y ROUTER DRAGINO.

Se realizó unas pruebas del simulador Diagslave y el router Dragino, con la finalidad de asegurar el correcto funcionamiento del protocolo Modbus en el router Dragino.

Teniendo el ejecutable del código, el cual es el encargado de interrogar al simulador Diagslave, que hace la función de un medidor de energía, en el router Dragino, se procedió a ejecutar el código write_uart.lua.

Como primer paso se procede a la compilación del archivo C, dicho archivo interrogara al simulador sobre los datos de voltaje, corriente, potencia y otros

parámetros de energía para una fase de una línea de distribución, como ejemplo práctico.

```
marvin_cornejo@ubuntu:~/Escritorio/tesis_marvin/Dragino$
‘/home/marvin_cornejo/opt/toolchains-mips/buildroot-2011.11/host/usr/bin/mips-
linux-gcc sensor01.c –o sensor011 `pkg-config –libs –cflags libmodbus`
```

Figura D.3. Compilación cruzada utilizando herramienta Buildroot.

En la Figura D.3 se muestra la línea de comandos a seguir para la compilación del código sensor01.c, el cual generara el ejecutable sensor011, el cual es trasferido al router Dragino con el comando scp, como se muestra en la Figura D.4.

```
Marvin_cornejo@ubuntu:~/Escritorio/Dragino$ scp sensor011 root@192.168.255.1:/
```

I Figura D.4. Comando scp para copiar a router Dragino.

Se procede a la conexión de la pc, en la cual estará en funcionamiento el simulador Diagslave con el router Dragino, realizando los pasos siguientes:

1. Colocar la IP de la computadora bajo el rango de IP que se encuentra el router Dragino.

```
marvin_cornejo@ubuntu:~$ sudo ifconfig eth0 192.168.255.2/24 up
```

Figura D.5. Configuración de IP en la computadora.

2. Abrir una Terminal y ejecutar los siguientes comandos:

```
marvin_cornejo@ubuntu:~$ ssh root@192.168.255.1

password:
```

Figura D.6. Ejecución de ssh para acceso a router Dragino.

Se escribe la contraseña para el acceso al router, se hace la conexión con ssh, el cual brinda seguridad en la trasferencia de datos de la computadora hacia el router.

3. Una vez conectado el router Dragino, se ejecuta la siguiente línea.

```
root@dragino-1988ca:/# ./sensor011
0.000,0.000, 0.000,0.000,0.000,0.000, 0.000
root@dragino-1988ca:/#
```

Figura D.7. Ejecución de sensor011 en router Dragino.

Otra forma que se utilizó, es por medio de un script Lua, el cual se puede observar en la Figura 3.10 de capítulo 3 de dicho trabajo, obteniendo el mismo resultado.

D.3. ANALISIS DE RESULTADOS CON WIRESHARK.

En dicha prueba se utilizó otra herramienta ya descrita en el apéndice c de este trabajo, la cual muestra el comportamiento de los paquetes que se generan en el tráfico de red, entre la pc y el router Dragino.

Al iniciar Wireshark, como primer paso se procede a escoger la red a la que se realizara el análisis, como lo muestra la Figura D.8.

Figura D.8. Selección de red a analizar por medio de wireshark.

Seleccionada la red, que para el caso es la red con IP 192.168.255.2, la red a la cual pertenece la pc que se comunicara con el router Dragino. Teniendo el tráfico de paquetes, según la peticiones que realice el router Dragino el cual ejecuta protocolo Modbus en modo maestro a la pc que ejecuta el simulador Diagslave, que es un esclavo para el caso.

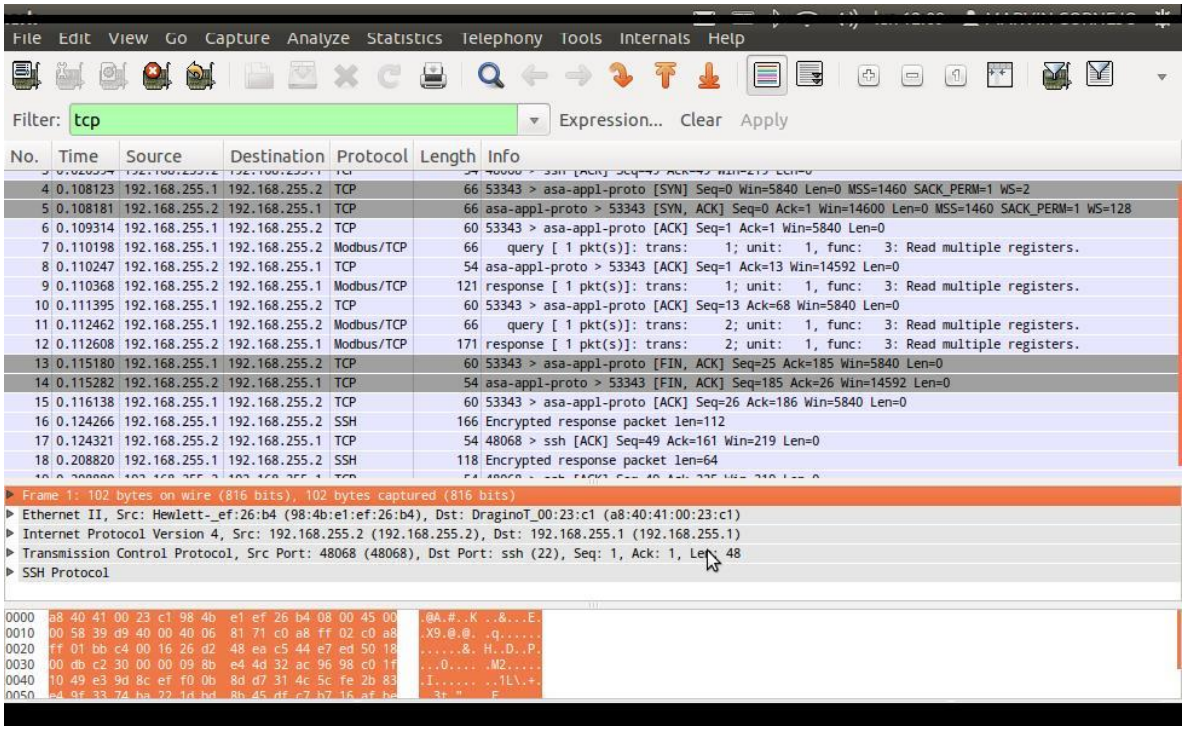


Figura D.9. Trafico de paquetes entre Dragino y pc.

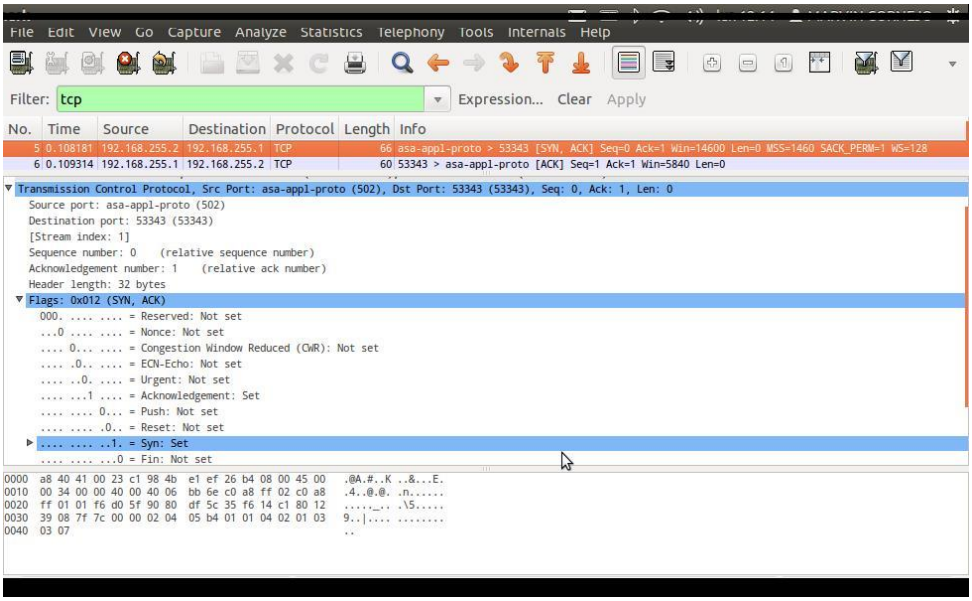


Figura D.10. Petición de sincronización de Dragino con pc.

Como se observa en la figura D.9, al inicio de la conexión tanto la fuente como el destino piden permiso de acceso a los puertos Ethernet, en el router Dragino que es la fuente, el puerto 502, configurado en el código C, el cual está bajo el protocolo Modbus y la computadora que es el destino con el puerto 53343.

Una vez completado el tráfico de paquetes entre la fuente y el destino, se procede al cierre de la conexión.

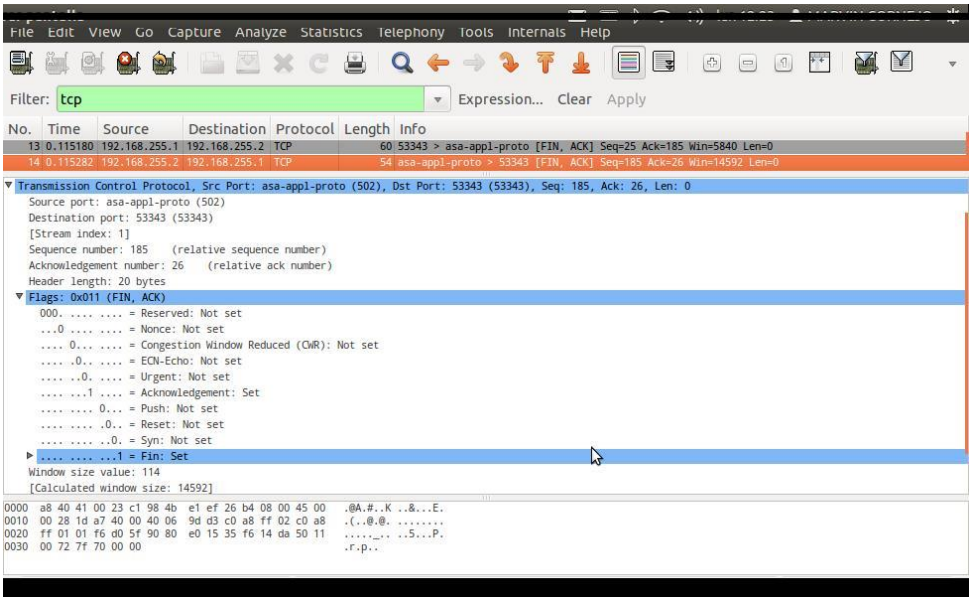


Figura D.11. Cierre de conexión.

Si se quiere tener detalles del comportamiento del tráfico de paquetes que se da entre el router Dragino y el simulador Diagslave en la pc, se utiliza la herramienta Flow Graph, la cual se encuentra en la barra de tareas de Wireshark, en la pestaña de Statistics.

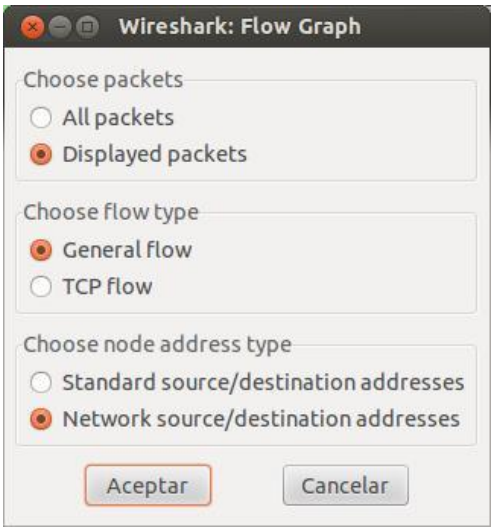


Figura D.12. Sub menu Flow Graph.

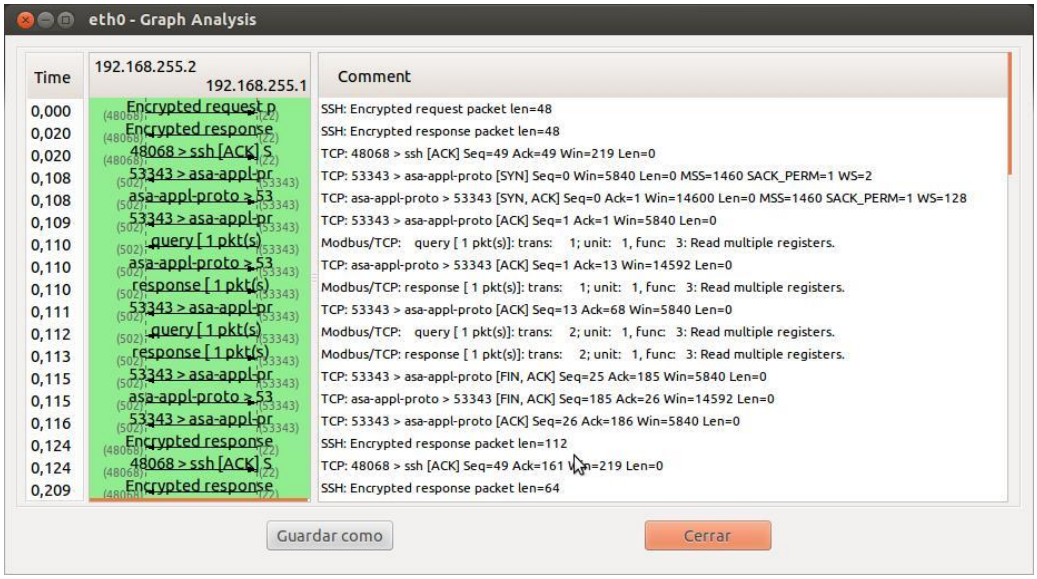


Figura D.13. Grafico del tráfico de paquetes entre simulador Diagslave y router Dragino.

BIBLIOGRAFIA.

[1] Datos sobre router Dragino. (Revisado 12/12/13).

http://www.dragino.com/about/about.html.

[2] Información básica de Arduino. (Revisado 12/12/13).

http://arduino.cc/en/Main/ArduinoBoardUno

[3] Información sobre compilación cruzada. (12/12/13).

http://linuxemb.wikidot.com/tesis-c3

[4] Buildroot (Revisado 12/12/13).

http://buildroot.uclibc.org/

[5] Referencia a Libmodbus (Revisado 12/12/13).

http://libmodbus.org/documentation/

[6] Puerto UART (Revisado 12/12/13).

http://www.zator.com/Hardware/H2_5_1_1.htm

[7] Lenguaje Lua. (Revisado 12/12/13).

http://www.lua.org/manual/5.1/es/manual.html#2.5.8

[8] Conexión entre Dragino y Arduino. (Revisado 12/12/13).

http://wiki.dragino.com/index.php?title=Dragino_UART

[9] Arquitectura MIPS. (Revisado 12/12/13).

http://es.wikipedia.org/wiki/MIPS_(procesador)

[10] Gestión de memoria en Linux. (Revisado 12/12/13).

http://es.wikipedia.org/wiki/Gesti%C3%B3n_de_memoria.

[11] AVR libc. (Revisado 12/12/13).

http://www.nongnu.org/avr-libc/user-manual/index.html

[12] Puerto Serial en Arduino. (Revisado 12/12/13)

http://arduino.cc/es/Reference/serial

[13] Librería SPI en Arduino. (Revisado 12/12/13).

http://arduino.cc/en/Reference/SPI

[14] Protocolo SPI. (Revisado 12/12/13).

http://es.wikipedia.org/wiki/Serial_Peripheral_Interface

[15] Buildroot para Libmodbus. (Revisado 12/12/13).

http://lwn.net/Articles/470996/

[16] Librerías Estáticas y Dinámicas. (Revisado 12/12/13).

http://luzem.dyndns.org/2009/10/18/librerias-estaticas-y-librerias-dinamicas/

[17] Librería uClib para sistemas embebidos. (Revisado 12/12/13).

http://www.uclibc.org/FAQ.html#naming

[18] librería SD.h. (Revisado 12/12/13).

http://arduino.cc/en/Reference/SD

[19] configuración de router como super nodo. (Revisado 22/02/2014)

Guía para la configuración de red malla usandi router Dir-300.pdf. Carlos Molina.

[20] crontab. (Revisado 22/02/2014)

http://www.linuxtotal.com.mx/?cont=info_admon_006

[21] cambio de capacidad de buffer puerto serial en Arduino. (revisado 22/02/2014)

http://www.hobbytronics.co.uk/arduino-serial-buffer-size

[21] Sistema Operativo OpenWrt. (Revisado 12/12/13).

https://openwrt.org/

[22] Protocolo ssh. (Revisado 12/12/13).

http://www.openssh.com/es/

[23] RTClib. (Revisado 12/12/13).

https://github.com/adafruit/RTClib.

[24] Función Modbus_new(). (Revisado 12/12/13).

http://libmodbus.org/site_media/html/modbus_new_tcp.html

[25] Función Modbus_connect().(revisado 12/12/13).

http://libmodbus.org/site_media/html/modbus_connect.html

[26] Función modbus_free().(revisado 12/12/13).

http://libmodbus.org/site_media/html/modbus_free.html

[27] Función modbus_close().(revisado 12/12/13).

http://libmodbus.org/site_media/html/modbus_close.html

[28] Función modbus_read_registers().(revisado 12/12/13).

http://libmodbus.org/site_media/html/modbus_read_registers.html

[29] Modelo cliente-servidor en una red TCP/IP. (revisado 12/12/13).

http://www.saulo.net/pub/tcpip/b.htm

[30] Introducción a Wireshark. (Revisado 12/12/13).

http://wiresharkdownloads.riverbed.com/video/wireshark/introduction-to-wireshark/

[31] Biblioteca Lua Socket. (Revisado 12/12/13).

http://w3.impa.br/~diego/software/luasocket/old/luasocket-2.0-alpha/introduction.html.

[32] Medidor Shark. (Revisado 12/12/13).

http://www.electroind.com/pdf/sp/ES149701_SP_Shark200_manual.pdf

[33] Simulador Diagslave. (Revisado 12/12/13).

http://www.modbusdriver.com/diagslave.html