ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN PARA RELÉS Y MEDIDORES A TRAVÉS DE SOFTWARE LIBRE SCADA PARA LA S/E 1 DE LA EERSA"

TESIS DE GRADO

Previa a la obtención del título de:

INGENIERO EN ELECTRÓNICA Y COMPUTACIÓN

Presentado por:

Wilmer Orlando Guamaní Proaño

Francisco Javier Narváez Quishpi

RIOBAMBA - ECUADOR

2011

A. Ing. Pablo Guevara, como Tutor de Tesis.

A. Ing. Jhonny Vizuete Miembro de Tribunal de Evaluación.

A. Ing. Jaime Guamaní Proaño por su valioso aporte.

A. Ing. Edwin Guamaní Proaño por su valioso aporte.

Infinitas gracias a mis padres y hermanos.

A toda mi familia.

A la Escuela Superior Politécnica de Chimborazo.

A la Empresa Eléctrica Riobamba S.A.

Por sobre todo a Dios.

Gracias.

Wilmer Guamaní Proaño.

A nuestro Señor Jesucristo.

A Sr. Jaime Ernesto Narváez Ortega por su apoyo y entrega incondicional.

A Sra. Rosa María Quishpi Chafla por su gran apoyo.

A Ing. Pablo Guevara, tutor de Tesis.

A Ing. Jhonny Vizuete, miembro de tribunal.

A Ing. Cesar Cepeda por su gran colaboración.

A Ing. Jorge Narváez, por su valioso aporte,

A Tlgo. Felipe Narváez, por su valioso aporte.

A Srta. Jacqueline Auquilla, por su gran apoyo.

A toda mi familia y hermanos, por su colaboración incondicional.

A la Escuela Superior Politécnica de Chimborazo

A la Empresa Eléctrica Riobamba S.A.

A todos los que participaron de una u otra manera

Gracias

Francisco Narváez.

A Dios por darme salud y vida.

A quienes han estado junto a mi durante toda la vida: mi Familia; mis amados padres: Gonzalo y Gloria; mis queridos hermanos: Jaime, Edwin, David Guamaní Proaño y amig@s.

Wilmer Guamaní Proaño.

"Nunca tomen el estudio como una obligación, sino como la envidiable oportunidad de aprender a conocer la influencia liberadora que ejerce la belleza sobre el Reino del Espíritu"

A. Einstein.

A mi Señor y Salvador Jesucristo, quien ha estado conmigo siempre y me ha enseñado como vivir en la vida.

A los que me acompañaron toda la vida con apoyo incondicional en todo momento: Mis padres, hermanos, familia en general, novia y amigos.

Francisco Narváez.

“Todo lo puedo en Cristo que me fortalece”

Filipenses 4:13

NOMBRE	FIRMA	FECHA
ING. IVAN MENES CAMEJO DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA	_____	
ING. PEDRO INFANTE DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES	_____	
ING. PABLO GUEVARA DIRECTOR DE TESIS	_____	
ING. JHONNY VIZUETE MIEMBRO DEL TRIBUNAL	_____	
TEC. CARLOS RODRÍGUEZ DIRECTOR CENTRO DE DOCUMENTACIÓN	_____	
NOTA DE LA TESIS	_____	

"Nosotros WILMER ORLANDO GUAMANÍ PROAÑO y FRANCISCO JAVIER NARVÁEZ QUISHPI, somos responsables de las ideas, doctrinas y resultados expuestos en esta: Tesis, y el patrimonio intelectual de la misma pertenecen a la Escuela Superior Politécnica de Chimborazo"

ÍNDICE DE ABREVIATURAS

SCADA: Supervisory Control And Data Adquisition.

MTU: Master Terminal Station.

RTU: Remote Terminal Unit.

HMI: Human Machine Interface.

PLC: Controlador Lógico Programable.

PAC: Controlador de Automatización Programable.

EERSA: Empresa Eléctrica Riobamba S.A.

S/E: Subestación.

TCP: Protocolo de Control de Transferencia.

IP: Protocolo de Internet.

UDP: Protocolo de Datagramas de Usuarios.

FTP: Protocolo de Transferencia de Archivos.

CONELEC: Consejo Nacional de Electricidad

DCS: Distributed Control System.

LAN: Local Area Network.

WAN: World Area Network.

DC: Corriente Directa.

DB: Database.

GPL: General Public License.

PC: Computador Personal.

RPY: Librería Python para Comunicación con el lenguaje estadístico R.

THD: Distorsión Armónica Individual.

IEEE: Institute of Electrical and Electronics Engineers.

DI: Entradas Digitales.

DO: Salidas digitales.

AC: Corriente Analógica.

LCD: Display de Cristal Líquido.

IETF: Internet Engineering Task Force.

CRC: Cyclic redundacy check.

ID: Identificador.

DAQ: Data Adquisition.

SFTP: SSH File Transfer Protocol.

CPU: Central Processing Unit.

SMS: Short Message Service.

ANSI: American National Standards Institute.

DNP: Distributed Network Protocol.

IEC: International Electrotechnical Commission.

RTD: Resistance Temperature Detectors.

ASCII: Stands for American Standard Code for Information Interchange.

DES: Data Encryption Standard.

DSA: Digital Signature Algorithm.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE GENERAL

CAPÍTULO II

SCADA

CAPÍTULO III

ESTUDIO COMPARATIVO SCADA

CAPÍTULO IV

LIKINDOY

CAPÍTULO V

DESCRIPCIÓN DEL EQUIPAMIENTO

CAPÍTULO VI

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE MONITORIZACIÓN DE LOS MEDIDORES Y RELÉS DE LA S/E1 DE LA EERSA

ANTECEDENTES

SCADA viene de las siglas de "Supervisory Control And Data Adquisition". Se trata de una aplicación de software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. Además, provee de toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa: control de calidad, supervisión, mantenimiento, etc.

Los tres componentes de un sistema SCADA son:

- ✓ Múltiples Unidades de Terminal Remota (RTU Remote Terminal Unit o Estaciones Externas).
- ✓ Estación Maestra y Computador con HMI (Human Machine Interface).
- ✓ Infraestructura de Comunicación.

La RTU se conecta al equipo físicamente y lee los datos de estado como los estados abierto/cerrado desde una válvula o un interruptor, lee las medidas como presión, flujo, voltaje o corriente. Por el equipo la RTU puede enviar señales que pueden controlarlo.

La RTU puede leer el estado de los datos digitales o medidas de datos analógicos y envía comandos digitales de salida o puntos de ajuste analógicos.

Una de las partes más importantes de la implementación de SCADA son las alarmas. Una alarma es un punto de estado digital que tiene cada valor NORMAL o ALARMA.

La alarma se puede crear en cada paso que los requerimientos lo necesiten. Un ejemplo de una alarma es la luz de "tanque de combustible vacío “del automóvil. El operador de SCADA pone atención a la parte del sistema que lo requiera, por la alarma. Pueden enviarse por correo electrónico o mensajes de texto con la activación de una alarma, alertando al administrador o incluso al operador de SCADA.

El término "Estación Maestra" se refiere a los servidores y al software responsable para comunicarse con el equipo del campo (RTU’s, PLCs, etc.) en estos se encuentra el software HMI corriendo para las estaciones de trabajo en el cuarto de control, o en cualquier otro lado. En un sistema SCADA pequeño, la estación maestra puede estar en un solo computador, A gran escala, en los sistemas SCADA la estación maestra puede incluir muchos servidores, aplicaciones de software distribuido, y sitios de recuperación de desastres.

El sistema SCADA usualmente presenta la información al personal operativo de manera gráfica, en forma de un diagrama de representación. Esto significa que el operador puede ver un esquema que representa la planta que está siendo controlada.

Los diagramas de representación pueden consistir en gráficos de líneas y símbolos esquemáticos para representar los elementos del proceso, o pueden consistir en fotografías digitales de los equipos sobre los cuales se animan las secuencias.

Los bloques software de un SCADA (módulos), permiten actividades de adquisición, supervisión y control.

Una interfaz Hombre - Máquina o HMI ("Human Machine Interface") es el aparato que presenta los datos a un operador (humano) y a través del cual éste controla el proceso.

Los sistemas HMI podemos pensarlos como una "ventana de un proceso". Esta ventana puede estar en dispositivos especiales como paneles de operador o en un ordenador. Las señales del proceso son conducidas al HMI por medio de dispositivos como tarjetas de entrada/salida en el ordenador, PLC's (Controladores lógicos programables), PACs (Controlador de automatización programable), RTU (Unidad de Terminal Remota) o DRIVER's (Variadores de velocidad de motores). Todos estos dispositivos deben tener una comunicación que entienda el HMI.

Los sistemas SCADA tienen tradicionalmente una combinación de radios y señales directas seriales o conexiones de módem para conocer los requerimientos de comunicaciones, incluso Ethernet e IP sobre SONET (fibra óptica) es también frecuentemente usada en sitios muy grandes como ferrocarriles y estaciones de energía eléctrica. Es más, los métodos de conexión entre sistemas pueden incluso que sea a través de comunicación wireless (por ejemplo si queremos enviar la señal a una PDA, a un teléfono móvil, etc.) y así no tener que emplear cables.

Para que la instalación de un SCADA sea perfectamente aprovechada, debe de cumplir varios objetivos:

- ✓ Deben ser sistemas de arquitectura abierta (capaces de adaptarse según las necesidades de la empresa).
- ✓ Deben comunicarse con facilidad al usuario con el equipo de planta y resto de la empresa (redes locales y de gestión).
- ✓ Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware. También tienen que ser de utilización fácil.

SCADA es popular debido a esta compatibilidad y seguridad. Ésta se usa desde aplicaciones pequeñas, como controladores de temperatura en un espacio, hasta aplicaciones muy grandes como el control de plantas nucleares.

La Empresa Eléctrica Riobamba S.A (EERSA), es la principal proveedora de energía en la provincia de Chimborazo. Los principales accionistas de la EERSA son los municipios de la provincia de Chimborazo, y el fondo de Solidaridad.

El principal proveedor de Energía Eléctrica para la EERSA son las centrales de Generación de Energía, siendo la principal La Hidroeléctrica de Paute, mediante la red de Interconectado Nacional.

La EERSA cuenta con algunas subestaciones ubicadas estratégicamente en toda la provincia de Chimborazo. En la ciudad de Riobamba la EERSA cuenta con algunas de estas Subestaciones. Entre ellas se encuentra la subestación 1, ubicada en la ciudad de Riobamba, entre la avenida “La Circunvalación” y la calle España. La Subestación 1 cuenta con 6 alimentadores principales, los cuales proveen de energía a distintos sectores de la ciudad, más un alimentador de reserva. Esta recibe energía mediante la red de Interconectado Nacional a un voltaje de 69000V (voltios), mediante el uso de transformadores se disminuye a un voltaje de 13800V (voltios), y mediante el uso de transformadores menos capacidad se obtiene los niveles de voltaje nominales para la distribución en la ciudad.

Cada alimentador cuenta con 4 relés de protección más un medidor digital ION 7300, cabe recalcar que además de eso la Subestación 1 cuenta con un armario central, en el cual se pueden obtener los valores de consumo total de todos los alimentadores, este armario cuenta con un medidor ION 7650.

El ION 7300 es un reemplazo ideal para un arreglo completo de medidores analógicos. Ofrece más de cien mediciones de energía, potencia, demanda y de calidad de energía y un puerto Ethernet opcional. Se puede integrar fácilmente el ION 7300 con sistemas de monitoreo mediante el protocolo propietario, por Modbus o por DNP3, compartiendo información a departamentos diferentes y a grupos de usuarios.

Los analizadores de redes ION7650 se utilizan en puntos clave de distribución y cargas sensitivas, ofrecen una funcionalidad inigualable que incluye análisis avanzados de la calidad de energía, alta precisión, opciones múltiples de comunicación, compatibilidad con la web y capacidades de control. Con su display gráfico de alta resolución, incluyen una extensa selección de pantallas configurables con una navegación sencilla y funcional. El ION 7650 viene con una extensa selección de pantallas de datos y mediciones pre configurada de forma que usted pueda encontrar la configuración requerida o personalizarla a sus requerimientos especiales. Se puede integrar el medidor con cualquier software u otro sistema de administración de energía o sistema SCADA mediante múltiples puertos y protocolos de comunicación.

Los relés de tipo IFCV51a es un relé monofásico de rango extendido, de tiempo inverso, de sobreintencidad de corriente, este tipo de relé tiene las siguientes características:

Activación: Es la corriente requerida para cerrar la posición de los contactos desde la posición de 0,5 en la caratula de tiempo.

Precisión de Tiempo de Operación: Estos funcionan dentro ±7% o 0,050 segundos, aunque este tiempo se puede ajustar según la necesidad.

Unidad Instantánea: Esta se refiere a uno de los métodos de protección que ofrece este relé ante fallas graves en la red protegida, como cortocircuitos permanentes, etc.

Unidad Temporizada: Es el segundo tipo de protección que el relé ofrece a la red ante fallas leves, como cortocircuitos esporádicos, subidas instantáneas en el voltaje, etc.

Por tanto, se implementara en la EERSA sistema SCADA para la supervisión de Relés y Medidores.

JUSTIFICACIÓN

El sistema pretende demostrar que tanto relés y medidores de la EERSA pueden ser supervisados remotamente mediante el sistema SCADA, el cual además permitirá saber el estado de funcionamiento de los equipos mencionados a tiempo real, a través de la emisión de reportes realizados según la necesidad de cada supervisor del sistema, estos reportes serán enviados utilizando las tecnologías existentes en la empresa, mediante la red y basándose en protocolos como el Modbus TCP/IP, o protocolos utilizados en la empresa.

A través del uso de la interfaz HMI, podemos proporcionar a los supervisores gráficas de gran calidad acerca de los estados de los equipos, por tanto garantizamos una rápida comprensión del estado de los mismos, además podemos publicar todos los datos que los supervisores requieran.

Como cualquier sistema SCADA, este sistema también funcionará a tiempo real el cual permitirá al supervisor del equipo realizar las acciones que precise en los equipos o comunicar estos daños a los respectivos técnicos, así evitando daños en los mismos, ahorrando tiempo, dinero y calidad en el servicio y mantenimiento de los equipos.

Todo el sistema estará implementado en el sistema operativo LINUX en la versión de Software que se ajuste a las necesidades del mismo, el cual nos brinda poderosas herramientas en cuanto a fiabilidad en los servicios ofrecidos, como la supervisión de datos, reportes, acciones en los dispositivos, etc., ya que LINUX está basado en el Sistema UNIX y por tanto es un sistema totalmente confiable.

OBJETIVOS

OBJETIVO GENERAL

Diseñar e implementar un sistema SCADA para la monitorización de Relés y Medidores de la S/E 1 de la Empresa Eléctrica Riobamba S.A.

OBJETIVOS ESPECÍFICOS

- ✓ Analizar los parámetros que se requieren para realizar una óptima supervisión de los relés y medidores de la S/E 1 de la EERSA.
- ✓ Investigar y realizar las conexiones necesarias de los dispositivos que se requieren para la monitorización de los relés y medidores de la S/E 1 de la EERSA.
- ✓ Analizar la solución más adecuada para la implantación del software SCADA entre todas las soluciones de software libre existentes en el mercado.
- ✓ Investigar la forma más efectiva y adecuada para la transmisión y recepción de datos que se requiere para monitorización de los relés y medidores desde los dispositivos hasta las estaciones de administración.
- ✓ Realizar el diseño de los reportes de acuerdo a los requerimientos de los supervisores de la S/E 1 de la EERSA.

HIPÓTESIS

La implementación de un Sistema de Monitorización SCADA a través de una solución de software libre para relés y medidores de la EERSA, permitirá la supervisión del funcionamiento de los equipos y además optimizar costos para la implementación de este tipo de sistemas.

CAPÍTULO I

INFORMACIÓN DE LA EMPRESA

1.1 Presentación

1.1.1 Filosofía

Es la de satisfacer toda demanda de electricidad que le sea requerida, ofreciendo a los clientes un servicio de calidad, con eficiencia y eficacia.

1.1.2 Visión

Empresa Distribuidora de energía eléctrica, al servicio de la provincia y de sus clientes, suministrando su energía, en las mejores condiciones técnico-económicas, preocupada permanentemente en la calidad del servicio y en los niveles de productividad.

1.1.3 Misión

Contribuir al desarrollo y satisfacción de los clientes, suministrando energía eléctrica con el esfuerzo, capacidad y motivación de su personal, implementando la más moderna tecnología, a fin de garantizar las necesidades y requerimientos presentes y futuros las veinticuatro horas del día, en condiciones técnicas y económicas a través del uso eficiente de la energía eléctrica.

1.2 Historia

En 1903 se funda la primera Sociedad que se encargaría de comercializar energía: "Alberto Rhor y Cía." que quiebra en 1907. En 1911 se funda la sociedad Anónima Riobamba Electric Ligth and Power. En 1924 se crea "La Hidroeléctrica" siendo uno de los socios Hirman Foley, apoderado de la Empresa Eléctrica del Ecuador inc., empresa que se encargó algunos años de la distribución de energía eléctrica. En 1953 se conforma la Empresa de Electrificación Chimborazo S. A. que tenía algunos proyectos como la construcción de la Central Hidroeléctrica Alao, algunas de cuyas obras estuvieron listas para el 20 de abril de 1961, fecha en la cual el Presidente de la República Dr. José María Velasco Ibarra, hizo su visita oficial.

El 3 de abril de 1963, nace la Empresa Eléctrica Riobamba S.A., quien compra todos los derechos a la Empresa de Electrificación Chimborazo S.A. y para el 2 de enero de 1967 realiza la inauguración de los dos primeros – grupos de la Central Alao, con la presencia del Dr. Otto Arosemena Gómez, Presidente de la República. En el año 1977 se inaugura el tercer grupo y para 1979 el cuarto y último grupo.

Para entonces, en 1972 y 1974 se habían adquirido grupos térmicos Ruston. y en 1976 la EERSA se había fusionado con la Empresa Eléctrica Alausí que contaba con una Central Hidroeléctrica llamada Nízag de 300 KW y en 1979 se pasó a formar parte del Sistema Nacional Interconectado, para luego iniciar la construcción de la línea San Juan - Alausí y las subestaciones San Juan, Guamote y Alausí, cada una con 1 MVA.

En 1977 y 1978 se compran los grupos ALCO (2.000 KW) y LISTER (457 KW). Se construye la línea de interconexión entre las subestaciones #s 1 y 2, además de la S/E #2 con 10 MVA, se inauguran en 1981 y en 1982 se construye la línea Alausí - Multitud - Pallatanga, lo que permitió electrificar hasta la zona límite con las provincias de Guayas y Bolívar.

Los Grupos térmicos General Motors, uno tipo estacionario y otro tipo paquete con 1800 Kw y 2000 Kw, respectivamente se adquieren en 1984 y en 1994 se tiene la enorme satisfacción de poner al servicio de la ciudad y Provincia la S/E#3 para posteriormente en 1995 poner en operación la línea Alausí-Chunchi con su respectiva subestación.

Para 1997, la EERSA inaugura la Central Hidroeléctrica Río Blanco con una potencia de 3 MW, con lo cual mejora notablemente el servicio a nuestra Ciudad y Provincia. Se electrificó nuestra ciudad, todos los Cantones de nuestra Provincia, muchas Comunidades y lugares inaccesibles por nuestra accidentada geografía.

En el año 2001 ingresa a formar parte del sistema eléctrico de la EERSA una nueva subestación denominada TAPI, la misma que toma la carga de la parte norte de la ciudad de Riobamba y parte del cantón Guano. A mediados del año 2002 la EERSA

cuenta con más de 108.000 abonados en toda la Provincia, el 68% de los cuales pertenecen al Sector Rural.

1.3 Sistema eléctrico EERSA

El Sistema Eléctrico de la Empresa Eléctrica Riobamba, se encuentra formado por el sistema de subtransmisión a 69 KV el cual está conectado al Sistema Nacional Interconectado, a través de la Subestación Riobamba perteneciente a TRANSELECTRIC.

El sistema de Subtransmisión consta de 10 subestaciones de distribución las cuales se describen en la tabla I.I.

Tabla I.I. Subestaciones de la EERSA.

Nombre de Subestación	Tipo E=Elevación R=Reducción S=Seccionamiento	VOLTAJE (kV)		CAPACIDAD DE LA SUBESTACIÓN (MVA)		
		1	2	OA	FA	FOA
S/E 1	R	69	13,8	10	12	
S/E2	R	69	13,8	10	12	
S/E 3	R	69	13,8	10	12	
S/E 4	R	69	13,8	10	12	
S/E 13 ALAO	R	69	13,8	2,5	3,125	
S/E 6 SAN JUAN CHICO	S	69				
S/E 7 CAJABABAMBA	R	69	13,8	2,5		
S/E 8 GUAMOTE	R	69	13,8	2,5		
S/E 9 ALAUSI	R	69	13,8	5		
S/E 10 CHUNCHI	R	69	13,8	1		

FUENTE: http://eersa.itgo.com/obras.htm

Dentro del sistema de Subtransmisión y Distribución, se encuentran conectadas las centrales de Generación del sistema, con una potencia total instalada de 16,34 Kw Las centrales con sus características se describen en la tabla I.II.

Tabla I.II. Centrales de Generación del Sistema.

Central	Tipo Central	Unidad	MARCA		Voltaje Nominal (kV)	Voltaje de Interconexión (kV)	POTENCIA	
			Motor ó Turbina	Generador			Nominal (kW)	Efectiva (kW)
ALAO	Hidráulica	Grupo 1	Bell	Brown Boveri	2,40	44	2.600	2.500
ALAO	Hidráulica	Grupo 2	Bell	Brown Boveri	2,40	44	2.600	2.500
ALAO	Hidráulica	Grupo 3	Bell	Brown Boveri	2,40	69	2.600	2.500
ALAO	Hidráulica	Grupo 4	Bell	Brown Boveri	2,40	69	2.600	2.500
RIO BLANCO	Hidráulica		Hidro Bouvier	AVK	6,00	13,8	3.125	3.000
NIZAG	Hidráulica		Weil Kreis Soest	Brown Boveri	0,48	13,8	312	300
RIOBAMBA	Térmica MCI		General Motors	Electr Motive Division	4,16	13,8	2.500	2.000

FUENTE: http://eersa.itgo.com/obras.htm

En relación a los transformadores de Potencia instalados en cada una de las Subestaciones sus datos característicos se muestran en la tabla I.III.

Tabla I.III. Transformadores de Potencia instalados en cada una de las Subestaciones

Nombre de Subestación	Descripción Subestación	Nombre Transformador o Autotransformador	Marca	Devanados (#)	POTENCIA (MVA)		VOLTAJE (kV)		TIPO DE CONEXIÓN	
					OA	FA	(P)	(S)	(P)	(S)
S/E 1		TRAFO S/E 1	SIEMENS	3	10	12,5	69	13,8	DELTA	Y
S/E 1	GENERACION	TERMICO	G.E	3	3,125		13,8	4,16	DELTA	Y
S/E 1		SALIDA A4/1	GE	3	1,25		13,8	4,16	Y	Y
S/E 2		TRAFO S/E 2	WESTHINGHOSE	3	10	12	69	13,8	DELTA	Y
S/E 3		TRAFO S/E 3	G.E	3	10	12	69	13,8	DELTA	Y
S/E 4		TRAFO S/E 4	G.E	3	10	12	69	13,8	DELTA	Y
S/E 13 ALAO	DISTRIBUCION	TRAFO S/E 13	SIEMENS	3	2,5	3,125	69	13,8	DELTA	Y
ALAO	GENERACION	GRUPO 1	BROWN BOVERI	3	3,28		44,2	2,4	DELTA	Y
ALAO	GENERACION	GRUPO 2	BROWN BOVERI	3	3,28		44,2	2,4	DELTA	Y
ALAO	GENERACION	GRUPO 3	BROWN BOVERI	3	3,28		69	2,4	DELTA	Y
ALAO	GENERACION	GRUPO 4	BROWN BOVERI	3	3,28		69	2,4	DELTA	Y
ALAO	GENERACION	AUTO-TRAFO	GENERAL E_CTRIC	3	6,56		69	4[illegible]	Y	Y
S/E 7 CAJABAMBA	RURAL	TRAFO S/E 7	OSAKA	3	2,5		69	13,8	DELTA	Y
S/E 8 GUAMOTE	RURAL	TRAFO S/E 8	SHI-LIN ELECTRIC	3	2,5		69	13,8	DELTA	Y
S/E 9 ALAUSI	RURAL	TRAFO S/E 9	MITSUBISHI E_CTRIC	3	5		69	13,8	DELTA	Y
S/E 10 CHUNCHI	RURAL	TRAFO S/E 10	WESTHINGHOSE	3	1	1,12	66	13,8	DELTA	Y
RIO BLANCO	GENERACION		SIEMENS	3	3,33		13,8	6	DELTA	Y
NIZAG	GENERACION			3	0,72		13,8	0,48	DELTA	Y

FUENTE: http://eersa.itgo.com/obras.htm

La red de subtransmisión de 69 kV, tiene una longitud aproximada de 134 km, el mismo que está compuesto por un anillo que rodea la ciudad de Riobamba y para el servicio del sur de la provincia e interconexión con la central Alao, se extiende como un sistema

radial. En la tabla I.IV se describe cada una de las líneas que componen el sistema de Subtransmisión de la EERSA:

Tabla I.IV. Líneas que componen el sistema de Subtransmisión de la EERSA.

NOMBRE DE SUBESTACIÓN		Topología R=Radial A=Anillo	Voltaje (kV)	Longitud (km)	Circuitos (#)	CONDUCTOR DE FASE		CONDUCTOR DE GUARDIA		Fecha de inicio de Operación
S/E Salida	S/E Llegada					Tipo	Calibre	Tipo	Calibre	
TRANSELECTRIC SAL 2–	S/E 1	A	69	0,91	1	ACSR	397	ACERO GALV.	5/8	1991
				6,4	1	ACSR	336	ACERO GALV.	5/8	
S/E 1	S/E 3	A	69	4,1	1	ACSR	336	ACERO GALV.	5/8	1983
S/E 3	S/E 2	A	69	3,6	1	ACSR	336	ACERO GALV.	5/8	1983
S/E 2	S/E 4	A	69	6,7	1	ACSR	336	ACERO GALV.	5/8	1990
S/E 4	TRANSELECTRIC SAL 1	A	69	2,3	1	ACSR	336	ACERO GALV.	5/8	2000
TRANSELECTRIC SAL 3	SAN JUAN	R	69	0,91	1	ACSR	397	ACERO GALV.	5/8	1991
				6,6	1	ACSR	336	ACERO GALV.	5/8	
SAN JUAN	ALAUSI	R	69	71	1	ACSR	2/0	ACERO GALV.	5/8	1980
ALAUSI	CHUNCHI	R	69	14,54	1	ACSR	2/0	ACERO GALV.	5/8	1995
S/E 1	ALAO	R	69	17	1	ACSR	3/0	ACERO GALV.	5/8	1967

FUENTE: http://eersa.itgo.com/obras.htm

1.3.1 Area de Concesion

Como empresa generadora, distribuidora y comercializadora, la EERSA debe prestar el servicio público de energía eléctrica de acuerdo a las bases de exclusividad y las áreas geográficas designadas por el CONELEC, e incluidas en su contrato de concesión. Teniendo la obligación de satisfacer la demanda de suministro de electricidad a los usuarios que están dentro de su área de servicio, la misma que se encuentra en las figuras I.1 y I.2 de localización geográfica:

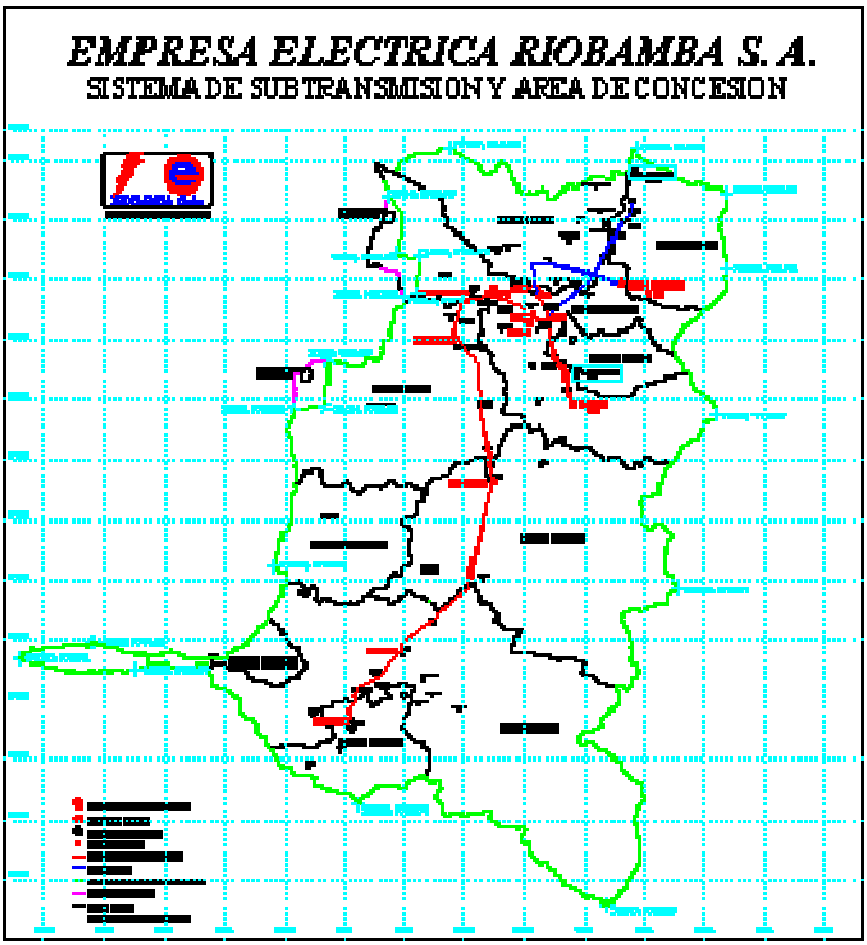


Figura I.1. Sistema de Subtransmisión de la EERSA

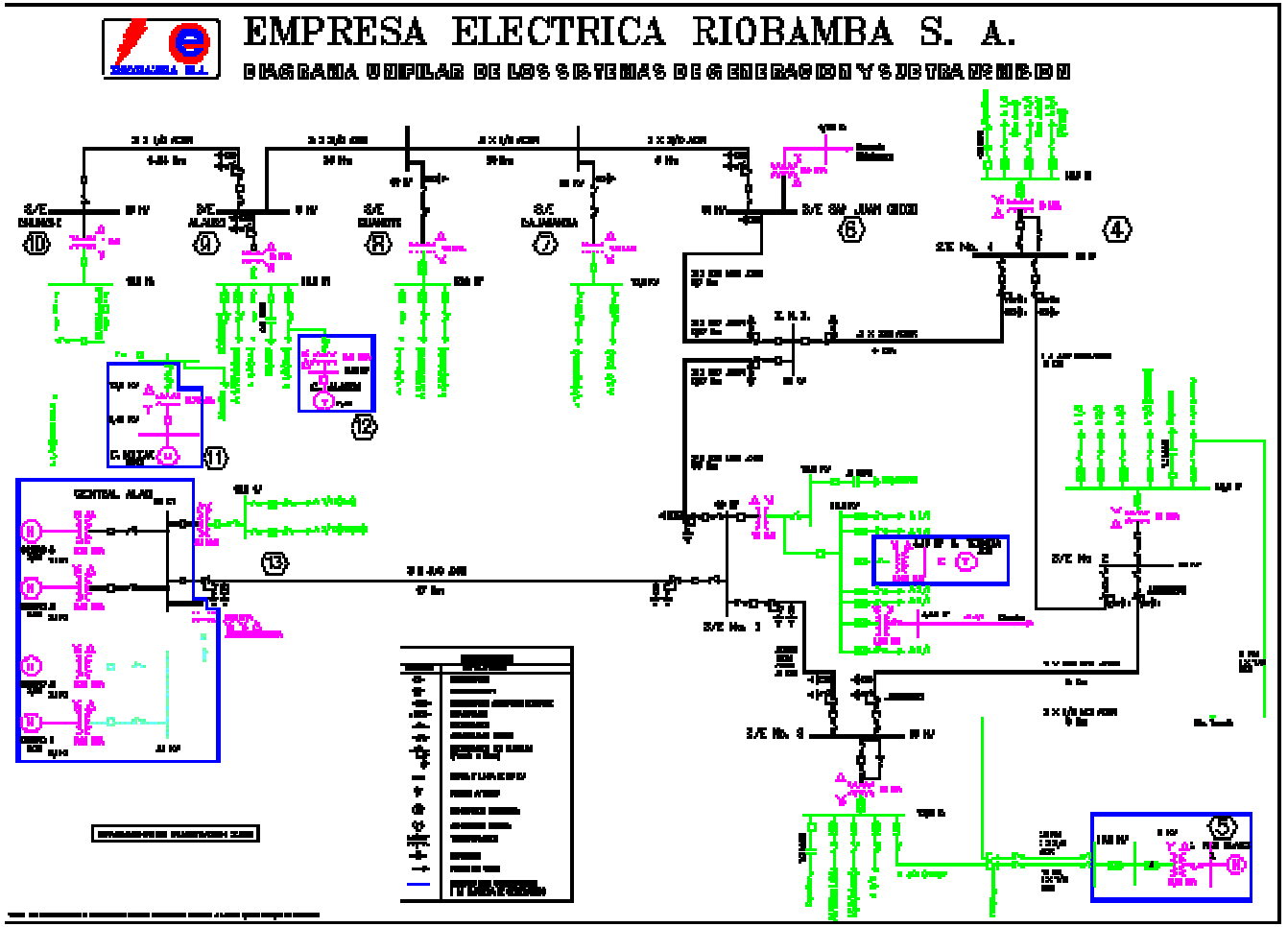


Figura I.2. Diagrama unifilar de los sistemas de generación y subtransmisión.

La cobertura del servicio que tiene la Empresa alcanza un 98,53% de la superficie total de la provincia de Chimborazo, sirviendo aproximadamente a 116.021 abonados distribuidos en 41% en el sector urbano (47.000 abonados) y 59% en el sector rural (69.021 abonados).

La Empresa Eléctrica Riobamba S.A. responsable en la continuidad, confiabilidad, calidad del producto y calidad del servicio; en base a un análisis de la relación causa – efecto y a la información proporcionada por el Departamento de Planificación y del Centro de atención al Cliente se realizó un muestreo de los centros de transformación a Intervenir y que servirá para realizar un diagnóstico, sobre la base del cual se determinarán los programas necesarios que permitan cumplir a cabalidad su responsabilidad con los clientes.

A Diciembre del 2001, en lo referente al sistema de Distribución, a través de las 10 Subestaciones de distribución se extienden 31 alimentadores primarios, con una longitud total de 2.644 km, de los cuales el 79% tiene configuración monofásica, el 18% es configuración trifásica y el 3% es bifásica.

En cuanto al número de transformadores de distribución y capacidad instalada en, la EERSA posee cuenta con un total de 6.842 transformadores de los cuales 6524 son monofásicos y la capacidad instalada es de un total de 122,826 MVA, de los cuales el 25% de la capacidad instalada corresponde a transformadores trifásicos.

En redes secundarias, se cuenta con una longitud total de 4.720 km de redes de baja tensión, de las cuales el 80% son circuitos monofásicos, los 19% bifásicos y el 1% son circuitos trifásicos.

1.4 Subestación No. 1 (CHIBUNGA)

1.4.1 Características

La subestación No 1 constituye una de las más importantes para la EERSA, por cuanto en ella convergen varias funcionalidades trascendentes para el sistema:

- ✓ Conforma el anillo de subtransmisión para la cuidad de Riobamba juntamente con las subestaciones No 2, No 3, No 4 y SNI.
- ✓ Es un punto de alimentación bidireccional de energía, central Alao – SNI.
- ✓ Posee generación térmica de 2.0 MVA.
- ✓ Es el centro de despacho de carga.
- ✓ Sera el centro supervisión y control del sistema de la EERSA.

1.4.2 Capacidad y Componentes

- ✓ Capacidad de transformación, 69kV a 13.8kV: 10MVA
- ✓ Capacidad de transformación, 13.8kV a 4.16kV: 3.125MVA
- ✓ Capacidad de generación a 4.16kV: 2.0MW
- ✓ Líneas de subtransmisión a 69kV

  Sistema Nacional Interconectado

  Central Alao

  Subestación S/E 3

- ✓ Alimentadores a 13.8kV:

  Alimentador A 1/1

  Alimentador A 2/1

  Alimentador A 3/1

  Alimentador A 5/1

  Alimentador A 6/1

- ✓ Alimentador a 4.16kV

  Alimentador A 4/1 (Chambo)

1.4.3 Ubicación y Facilidades

Está ubicado al lado occidental de la ciudad, en la Av. Circunvalación. Cuenta con suficiente espacio físico y por su lugar en el límite de la urbe, no presenta típicos de interferencia para las comunicaciones.

- ✓ Altura 2760 m.s.n.s.
- ✓ Latitud sur 1^{o} 40’ 42’’
- ✓ Longitud oeste 78^{o} 39’ 56’’

CAPÍTULO II

SCADA

2.1 Introducción

La evolución de los sistemas operativos ha incrementado tambien las posibilidades de estos sistemas, los lenguajes de programación gráfica iniciales se fueron adaptando a las nuevas posibilidades que ofrecia la técnica.

Los primeros paquetes de software para el control y adquisición de datos, SCADA, como Genesis (Iconics), utilizaban las capacidades graficas del lenguaje BASIC. SCADA es cualquier software que permita el acceso a datos remotos de un proceso y, permita, utilizando las herramientas de comunicación nesesarias en cada caso el control del mismo.

No se trata de un sistemas de control, sino de una utilidad de sofware de monitorizacion o supervicion, que realiza la tarea de interface entre niveles de control y los de gestion, a un nivel superior.

La topologia de un sistema SCADA (su distribución fisica) variará adecuandose a las caracteristicas de cada aplicación. Unos sistemas funcionaran bien en configuraciones de bus, otros en configuraciones de anillo, etc. Unos necesitan equipos redundantes debido a las caracteristicas de proceso, etc.

2.2 Conceptos Generales

2.2.1 Definición de un sistema SCADA

SCADA viene de las siglas de "Supervisory Control And Data Adquisition". Se trata de una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. Además, provee de toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa: control de calidad, supervisión, mantenimiento, etc.

En este tipo de sistemas usualmente existe un ordenador, que efectúa tareas de supervisión y gestión de alarmas, así como tratamiento de datos y control de procesos.

La comunicación se realiza mediante buses especiales o redes LAN. Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos.

Los programas necesarios, y en su caso el hardware adicional que se necesite, se denomina en general sistema SCADA.

Un SCADA no debe confundirse con un Sistema de Control Distribuido (DCS, Distributed Control System), aunque actualmente los principios y tecnologías que utilizan son muy similares. Su principal diferencia consiste en que los sistemas de control distribuido, normalmente se usan para controlar procesos industriales más complejos y restringidos al perímetro de una planta; por ejemplo, los sistemas de control de una refinería, los de una planta de GLP, etc. En la tabla II.V se muestra un cuadro comparativo de las principales características de los sistemas SCADA y los sistemas de Control Distribuido (DCS) (Estas Características no son limitantes para uno u otro tipo de sistemas, son típicas).

Tabla II.V. Algunas diferencias típicas entre sistemas SCADA y DCS.

ASPECTO	SCADA's	DCS
TIPO DE ARQUITECTURA	Centralizada	Distribuida
TIPO DE CONTROL PREDOMINANTE	SUPERVISORIO: Lazos de control cerrados por el operador. Adicionalmente: control secuencial y regulatorio.	REGULATORIO: Lazos de control cerrados automáticamente por el sistema. Adicionalmente: control secuencial, batch, algoritmos avanzados, etc.
TIPOS DE VARIABLES	Desacopladas	Acopladas

Tabla II.V. Algunas diferencias típicas entre sistemas SCADA y DCS. (Continuación)

ÁREA DE ACCIÓN	Áreas geográficamente distribuidas.	Área de la planta.
UNIDADES DE ADQUISICIÓN DE DATOS Y CONTROL	Remotas, PLCs.	Controladores de lazo, PLCs.
MEDIOS DE COMUNICACIÓN	Radio, satélite, líneas telefónicas, conexión directa, LAN, WAN.	Redes de área local, conexión directa.
BASE DE DATOS	Centralizada	Distribuida

FUENTE: Original.

El SCADA describe un número de unidades terminales remotas (RTU´s, Remote Terminal Units) instaladas en las cercanías del proceso, las cuales se comunican con una estación maestra (MTU, Master Terminal Station) ubicada en una sala de control central.

Una RTU es un sistema que cuenta con un microprocesador e interfaces de entrada y salida tanto analógicas como digitales que permiten tomar la información del proceso provista por los dispositivos de instrumentación y control en una localidad remota y utilizando técnicas de transmisión de datos enviarla al sistema centralizado maestro. La MTU bajo un software de control, permite la adquisición de los datos a través de todas las RTU's ubicadas remotamente y brinda la capacidad de ejecutar comandos de control remoto cuando es requerido por el operador. Los datos adquiridos por la MTU se

presenta a través de una interfaz gráfica en forma comprensible y utilizable, y más aun esta información puede ser impresa en un reporte.

2.2.2 Flujo de información en los sistemas SCADA.

En un proceso automatizado intervienen numerosas variables de proceso; dependiendo del fenómeno físico que se observe (presión, temperatura, flujo, etc); estos fenómenos físicos son captados por un transductor, el cual alimenta una señal eléctrica a un transmisor este transmisor entrega una señal análoga eléctrica en forma de voltaje o corriente normalizada de 4 a 20 [mA], o 0 a 5 [V] DC, o desde 0 a 10 [V] DC.

Estas señales eléctricas deben ser procesadas para que puedan ser transmitidas mediante técnicas digitales y eventualmente entendidas por una computadora, por lo que se necesita hacer una conversión de datos análogo/digital o viceversa. Luego todas las señales digitales se envían hacia un cuarto de control donde se reúne la información de toda la planta industrial. Simultáneamente se muestra la información en una pantalla del computador para que el operador pueda tomar decisiones; estos datos digitalizados son almacenados para su análisis, proporcionando así datos históricos para la toma de decisiones.

En aquellos lugares donde se debe manipular algunas variables durante el proceso de automatización y se encuentran distribuidas dentro de áreas extensas se requiere de una unidad terminal remota (RTU) este dispositivo permite concentrar la información de varios transductores/actuadores y luego son transmitidos hacia una estación maestra o unidad terminal maestra (MTU).

En la figura II.3. Se presenta un esquema básico de un sistema SCADA:

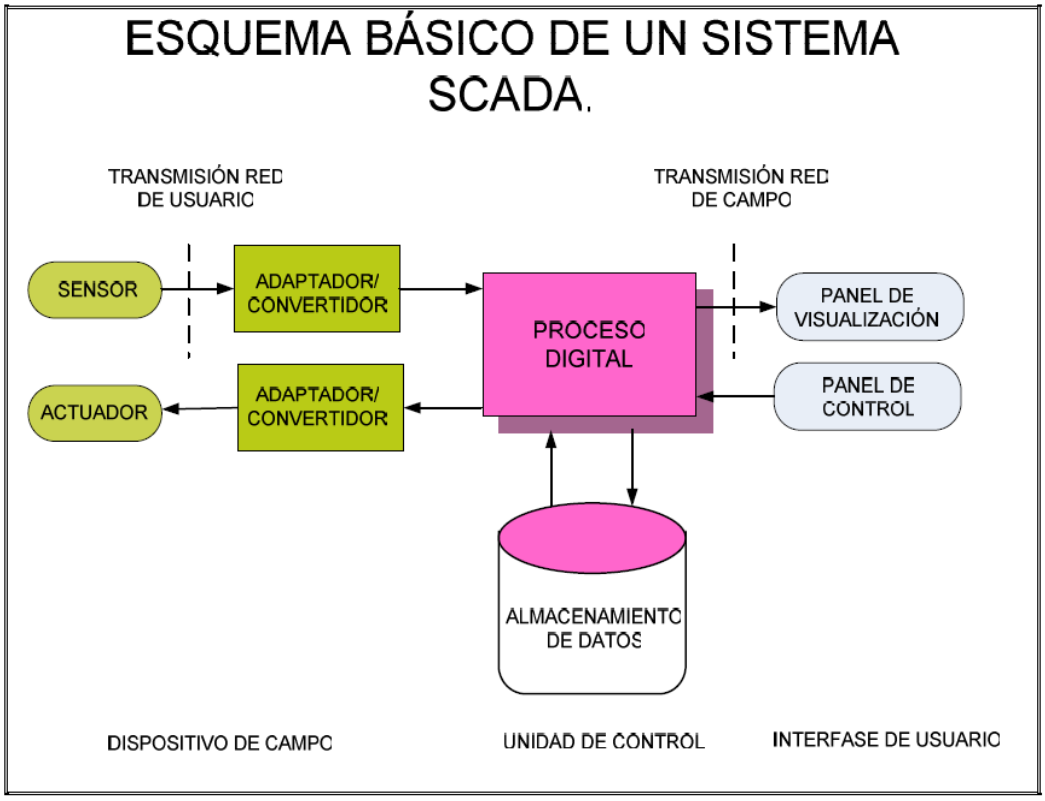


Figura II.3. Esquema básico de un Sistema SCADA.

2.2.3 Funciones principales del sistema

Dentro de las funciones básicas realizadas por el sistema SCADA están las siguientes:

- ✓ Supervisión remota de instalaciones y equipos: Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.

- ✓ Control remoto de instalaciones y equipos: Mediante el sistema se puede activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, prender motores, etc.), de manera automática y también manual.

- ✓ Además es posible ajustar parámetros, valores de referencia, algoritmos de control, etc.

- ✓ Procesamiento de datos: El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada, y comparada con datos anteriores, y con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.

- ✓ Visualización gráfica dinámica: El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente dentro de una planta real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.

- ✓ Generación de reportes: El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.

- ✓ Representación se señales de alarma: A través de las señales de alarma se logra alertar al operador frente a una falla o la presencia de una condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.

- ✓ Almacenamiento de información histórica: Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o del autor del programa.

- ✓ Programación de eventos: Esta referido a la posibilidad de programar subprogramas que brinden automáticamente reportes, estadísticas, gráfica de curvas, activación de tareas automáticas, etc.

2.2.4 Prestaciones.

Un paquete SCADA debe estar en disposición de ofrecer las siguientes prestaciones:

- ✓ La monitorización: Representación de datos en tiempo real a los operadores de planta. Se leen los datos de los autómatas (Temperaturas, velocidades, etc), posibilidad de vigilancia a muchos kilómetros de distancia.

- ✓ Supervisión, mando y adquisición de datos de un proceso y herramientas de gestión para la toma de decisiones (mantenimiento preventivo, etc). Tiene también la capacidad de ejecutar programas que puedan supervisar y modificar el control establecido y, bajo ciertas condiciones, anular o modificar tareas asociadas a los autómatas. Evitando una continua supervisión humana.

- ✓ Adquisición de Datos de los Procesos en Observación: Se puede observar mediante herramientas registradoras y obtener así un valor medio de la irradiación en la zona, guardando los valores obtenidos y evaluándolos a posterioridad.

- ✓ La visualización de los estados de las señales del Sistema (alarma y eventos): Reconocimiento de eventos excepcionales acaecidos en la planta y su inmediata

puesta en conocimiento a los operarios para efectuar las acciones correctoras pertinentes. Además, los paneles de alarma pueden exigir alguna acción de reconocimiento por parte del operario, de forma que queden registradas las incidencias.

- ✓ El mando: Posibilidad que los operadores puedan cambiar consignas u otros datos claves del proceso directamente desde el ordenador (marcha, paro, etc). Se escriben datos sobre los elementos de control.

- ✓ Grabación de acciones o recetas: En algunos procesos se utilizan combinaciones de variables que son siempre las mismas. Un sistema de recetas permite configurar toda una planta de producción ejecutando un solo comando. La línea de vulcanizado en continuo, se compone de varias maquinas encadenadas con múltiples parámetros (velocidad y temperatura principalmente), que dependen del tipo de perfil a elaborar.

- ✓ Garantizar la seguridad de los datos: Tanto envió y recepción de datos deben de estar suficientemente protegidos de influencias no deseadas, intencionadas o no (fallos en la programación, intrusos, situaciones inesperadas, etc).

- ✓ Garantizar la seguridad en los accesos: Restringiendo zonas de programa comprometidas a usuarios no autorizados, registrando todos los accesos y acciones llevadas a cabo por cualquier operador.

- ✓ Posibilidad de programación numérica: Permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador (Lenguajes de alto nivel, C y Visual Basic, etc.)

2.2.5 Requisitos.

Un SCADA debe cumplir varios objetivos para que su instalación sea perfectamente aprovechada:

- ✓ Deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa.

- ✓ Deben comunicarse con total facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).

- ✓ Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables con el usuario.

2.3 Elementos Básicos

2.3.1 Componentes de Hardware

Un sistema SCADA, necesita ciertos componentes inherentes de hardware en su sistema, para poder tratar y gestionar la información captada:

- ✓ Unidad terminal maestra (MTU).
- ✓ Unidad terminal remota (RTU).
- ✓ Red de comunicación.

- ✓ Instrumentación de campo.

2.3.1.1 Unidad Terminal Maestra (MTU).

La Unidad Terminal Maestra es el computador principal del sistema el cual supervisa y recoge la información del resto de las subestaciones; soporta una interfaz hombre máquina.

El sistema SCADA más sencillo es el compuesto por un único computador, el cual es la Unidad Terminal Maestra (MTU) que supervisa toda la estación.

2.3.1.1.1 Interfaz Hombre - Máquina (HMI)

Un sistema SCADA incluye un interfaz de usuario, generalmente llamado Human Machine Interface (HMI). El HMI de un sistema SCADA es donde los datos son procesados y presentados para ser visto y controlado por un operador humano. Esta interfaz incluye los controles por lo general en que el individuo puede interactuar con el sistema SCADA.

Las HMI son una manera fácil de estandarizar la facilitación de la supervisión de múltiples RTU’s o PLC’s controladores lógicos programables. Por lo general, RTU o PLC se ejecutará un proceso pre programado, pero la supervisión a cada una de ellas puede ser difícil, porque por lo general se extienden sobre el sistema. Debido al RTU y PLC tenido históricamente ningún método normalizado para mostrar o presentar los datos a un operador, el sistema SCADA se comunica con PLC's en toda la red del sistema y procesos de información que se disemina fácilmente por el HMI. HMI también puede estar vinculado a una base de datos, que pueden utilizar los datos recogidos desde el PLC o RTU para proporcionar gráficos sobre las tendencias,

información logística, esquemas de un sensor específico o de la máquina o incluso hacer que la solución de problemas guías accesibles. En la última década, prácticamente todos los sistemas SCADA HMI incluyen un sistema integrado y el dispositivo PLC lo que resulta extremadamente fácil de ejecutar y supervisar un sistema SCADA.

2.3.1.2 Unidad Terminal Remota (RTU).

Una Unidad Terminal Remota es un dispositivo instalado en una localidad remota del sistema está encargado de recopilar datos para luego ser transmitidos hacia la Unidad Terminal Maestra.

Esta unidad está provista de canales de entrada para detección o medición de las variables de un proceso y de canales de salida para control o activación de alarmas y un puerto de comunicaciones; físicamente estos computadores son tipo armarios de control.

Una tendencia actual es la de dotar a los PLCs (Controlador Lógico Programable) la capacidad de funcionar como unidad terminal remota.

2.3.1.3 Red de Comunicación.

El sistema de comunicación es el encargado de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA.

El tipo de medio utilizado en las comunicaciones puede variar según las necesidades del sistema y del programa del software escogido para implementar el sistema SCADA, ya que no todos los programas de software así como los instrumentos de campo pueden trabajar con el mismo medio de comunicación.

2.3.1.4 Instrumentación de Campo.

Los instrumentos de campo están constituidos por todos aquellos dispositivos que permiten tanto realizar la automatización o control del sistema (PLCs, controladores de procesos industriales, y actuadores en general) son los que se encargan de la captación de información del sistema.

2.3.2 Módulos de un SCADA.

Los módulos o bloques software que permiten las actividades de adquisición, supervisión y control son los siguientes:

- ✓ Configuración.
- ✓ Interfaz gráfico del operador.
- ✓ Módulo de proceso.
- ✓ Gestión y archivo de datos.
- ✓ Comunicaciones.

2.3.2.1 Configuración.

Permite al usuario definir el entorno de trabajo de su aplicación SCADA, dentro del módulo de configuración el usuario define las pantallas gráficas o de texto que va a utilizar, adaptándolo a la aplicación particular que se desea desarrollar.

2.3.2.2 Interfaz gráfico del operador.

Proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y

generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.

Permite al operador determinar el estado de los dispositivos de campo (prendido/apagado) que están presentes en los procesos industriales proporcionando al operador las funciones de control y supervisión de la planta.

2.3.2.3 Módulo de proceso.

Ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas.

2.3.2.4 Gestión y archivo de datos.

Se encarga del almacenamiento y procesado ordenado de los datos, según formatos inteligibles para periféricos: hardware (impresoras, registradores) o software (bases de datos, hojas de cálculo), etc. de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

2.3.2.5 Comunicaciones.

Se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de elementos informáticos de gestión.

2.4 Soluciones SCADA

A continuación en la tabla II.VI se citan algunas soluciones SCADA, las cuales son las más usuales en el mercado:

Tabla II.VI. Soluciones SCADA.

PRODUCTO	CARACTERÍSTICAS	FABRICANTES
Likindoy	SCADA libre, fuertemente basado en protocolos y tecnologías popularizadas en Internet. Intenta ser un sistema simple, Fácil de entender por e ingenieros de control.	OpenSource
Open SCADA Project	✓ Sistema abierto. ✓ Apertura (GPL), multiplataforma, módulos y la escalabilidad.	OpenSource
Labview	✓ Ambiente de desarrollo gráfico con una metodología muy fácil de dominar por ingenieros y científicos. ✓ Se pueden crear fácilmente interfaces de usuario para la instrumentación virtual sin necesidad de elaborar código de programación, para especificar las funciones sólo se requiere construir diagramas de bloque.	National Instruments
Lookout	✓ Software que permite crear fácilmente poderosas aplicaciones de monitoreo y control de procesos, el desarrollo de su interface hombre-máquina le toma menos tiempo permitiéndole ahorrar sustancialmente en el costo total.	National Instruments

Tabla II.VI. Soluciones SCADA. (Continuación).

Registro de Datos y Control Supervisorio de labview	✓ El Módulo NI LabVIEW Datalogging and Supervisory Control (DSC) es el complemento de LabVIEW, para desarrollar su HMI/SCADA o aplicaciones de registro de datos de muchos canales.	National Instruments
Sattgraph 5000	✓ Es un sistema modular SCADA/HMI abierto. Está basado en la técnica SattGraph y se ejecuta sobre Microsoft Windows.	ABB
The compact HMI 800	✓ Minimiza el esfuerzo realizado en el manejo de HMI para PLCs y otros tipos de controladores, de las operaciones, mantenimiento y perspectivas de la ingeniería.	ABB
Virgo 2000	✓ Diseñado para permitir ambos tipos de control dentro de un proceso industrial: ✓ Control automático, donde una computadora controla directamente el proceso de supervisión y control, donde un operador humano, dirige el equipo.	AlterSys Inc.
Proficy HMI/SCADA CIMPLICITY	✓ Solución cliente/servidor de visualización y control. ✓ Proporciona funcionalidades de visualización de procesos, adquisición de datos y control para entornos de proceso y de fabricación.	General Electric

Tabla II.VI. Soluciones SCADA. (Continuación).

Proficy HMI/SCADA - ifix	✓ Solución de gestión de la información en tiempo real. ✓ Abierta, flexible y escalable, e incluye sorprendentes funciones de visualización de nueva generación, un fiable motor de control, un potente sistema de históricos integrado.	General Electric
Génesis 64	✓ Diseñado y certificado para Microsoft Windows Vista. ✓ Aprovecha la tecnología de 64-bit de AMD e Intel, lo que permite un desarrollo más rápido de su solución de automatización. ✓ Es People-Ready, permite a los trabajadores de la planta y los profesionales de TI a integrar en tiempo real la información comercial.	Iconics
Génesis 32	✓ Software de quinta generación. ✓ Se basa en la tecnología OPC-To-The-Core y soporta múltiples plataformas de Microsoft.	Iconics
Daqfactory	✓ Control para la vigilancia y el control general de productos de automatización industrial. ✓ Permite controlar los procesos, puntos de ajuste de cambio, realice el registro y análisis de datos. ✓ Permite hacer, todo desde su PC de la oficina o un PC de planta por separado. Si está utilizando PLC para la automatización de fábrica.	Azeotech

Tabla II.VI. Soluciones SCADA. (Continuación).

SCADA System SIMATIC wincc	✓ Facilidad de usar. ✓ Con este sistema configurable y escalable, tiene la ventaja de la apertura absoluta tanto en el entorno de oficina y de producción. ✓ Base de datos de proceso integrado e Inteligencia Planta, por ejemplo, garantizar la transparencia en la producción. ✓ Numerosas opciones y add-ons extender y ampliar el ámbito de actuación	Siemens
SIMATIC winac ODK (Open Development Kit)	✓ Permite el uso flexible de todos los recursos del PC con el programa de control a través de tres diferentes interfaces. ✓ Todas las funciones del sistema operativo Windows y el sistema de resources of. ✓ Proporciona acceso al hardware y componentes de software externos.	Siemens

Tabla II.VI. Soluciones SCADA. (Continuación).

PROASIS DCS-Win y PROASIS DCS-Plus	✓ Software SCADA de Supervisión y Control de medidas y señales lógicas en PC. ✓ Entorno Windows. ✓ Dispositivos de adquisición de datos con comunicación por **RS-485 Modbus**, por **Radio** o por **Ethernet** cable o **WiFi**, para supervisión de procesos industriales. ✓ **PROASIS DCS-Plus** es un **SCADA** modular de Supervisión y Control para trabajar en redes LAN.	Desin-Instruments

Fuente: Original.

Como se puede observar en la tabla anterior, aleatoriamente se han escogido algunos software SCADA populares en el mercado. La mayoría de ellos son licenciados, por lo que no se tomaran en cuenta en este proyecto, ya que se pretende realizar el SCADA bajo software libre.

Por tanto, los únicos software que quedan a discusión son Likindoy y OpenScada. Existen otras soluciones SCADA's basadas en software libre, mas estas no se han tomado en cuenta, ya que siguen bajo desarrollo, y no tienen tantas prestaciones como los mencionados anteriormente.

CAPÍTULO III

ESTUDIO COMPARATIVO SCADA

3.1 Open SCADA

OpenSCADA es un sistema SCADA abierto principalmente construido en módulos, multiplataforma y escalables. SCADA es el término que se utiliza a menudo para describir la automatización de procesos tecnológicos. El sistema OpenSCADA está diseñado para: recoger, archivar, y visualizar la información, liberar operaciones influentes, y otras operaciones relacionadas, característico de todas las funciones de sistemas SCADA como se muestra en la figura III.4.

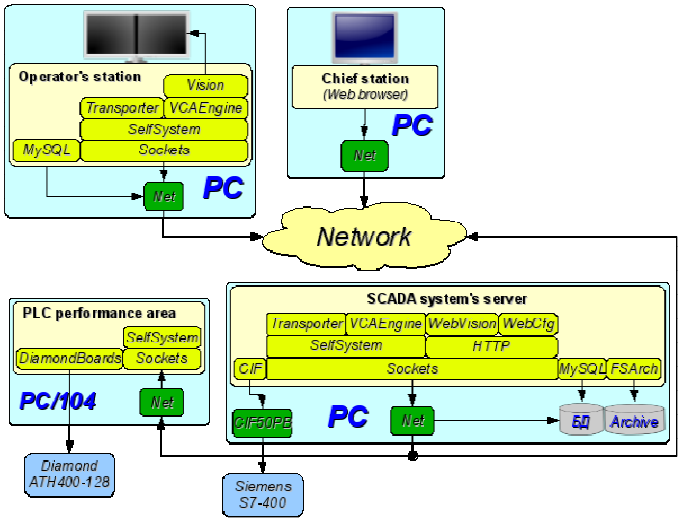


Figura III.4. Funciones del Sistema SCADA.

3.1.1 Alcances

El sistema está diseñado para alcanzar un rendimiento en sus funciones como los sistemas SCADA comunes, y para su uso en áreas adyacentes de las tecnologías de la información.

El sistema OpenSCADA puede ser utilizado en:

- ✓ Objetivos industriales, teniendo todas las funciones de un sistema SCADA.
- ✓ La construcción de un sistema, como las áreas de ejecución (incluyendo PLC).
- ✓ Para la construcción de varios modelos (tecnológicas, químicas, físicas, procesos eléctricos).
- ✓ En los ordenadores personales, servidores y los clusters para la recolección, procesamiento, representación y el archivo de la información en el sistema y su entorno.

Como base para el desarrollo y los usos, es elegido el sistema operativo Linux, que es un sistema operativo compatible con POSIX. Además el sistema operativo Linux cuenta con buena asistencia en el momento que tenemos preguntas sobre:

- ✓ Seguridad.
- ✓ Flexibilidad / Escalabilidad.
- ✓ Disponibilidad.
- ✓ popularidad y prevalencia.

Como el sistema OpenSCADA se desarrolla en sistema operativo estándar POSIX, ya que este tiene el beneficio de ser multiplataforma, y de esta manera no tener problemas al momento de adaptarse con un sistema operativo que no sea multi-plataforma.

3.1.2 Arquitectura

Es el núcleo modular del sistema. Para lograr alta velocidad en la comunicación debido a la reducción del tiempo, la arquitectura permite unir las funciones de los sistemas distribuidos en un programa.

Arquitectónicamente, el sistema OpenSCADA consiste en los siguientes subsistemas:

- ✓ El subsistema de seguridad.
- ✓ Los módulos de subsistema de DB (Data Base).
- ✓ El subsistema de transporte módulos.
- ✓ El transporte de los módulos de protocolo del subsistema.

- ✓ Los módulos del subsistema DAQ.
- ✓ El módulo de archivos del subsistema.
- ✓ Los módulos de interfaces de usuarios del subsistema.
- ✓ El control de módulos del Subsistema.
- ✓ Los módulos especiales del Subsistema.

Partiendo del principio de los módulos, los subsistemas modulares especifican que modulo puede ampliar la funcionalidad mediante la conexión de módulos del tipo correspondiente.

El núcleo modular del sistema OpenSCADA se maneja en forma de librerías estáticas y compartidas. Este permite construir en función de los programas existentes en el sistema, y también para crear nuevos programas sobre la base de un núcleo modular del sistema OpenSCADA.

Sin embargo, el núcleo modular es autosuficiente y se puede utilizar para iniciar un programa simple.

Módulos del sistema OpenSCADA se almacenan en librerías dinámicas. Cada librería dinámica puede contener un conjunto de módulos de varios tipos. La provisión de estas librerías dinámicas se define por la conectividad funcional de los módulos. Las bibliotecas dinámicas permiten el reemplazo en caliente que permite realizar la actualización de los módulos mientras están trabajando.

3.2 Likindoy

Likindoy es un SCADA libre, fuertemente basado en protocolos y tecnologías popularizadas en Internet. Intenta ser un sistema simple, fácil de entender.

Likindoy utiliza:

- ✓ Tecnología Open Source:
- ✓ El sistema operativo es Linux.
- ✓ El lenguaje de programación es Python.
- ✓ El gestor de bases de datos es MySQL

Para la generación de gráficas se usa el lenguaje de programación **R** ideado para uso estadístico y representación visual de datos.

Likindoy es una iniciativa de Axaragua, que lo usa como sistema propio de gestión. Es un sistema que puede ser muy útil a cualquiera que necesite automatizar, controlar y/o representar flujos de datos. Por ejemplo: la temperatura en una ciudad, el tráfico de red, el consumo de agua en una finca, etc.

Likindoy es un programa diseñado para análisis industrial y sistemas de telecontrol. Permite a otros programadores ampliar su funcionamiento con facilidad.

Likindoy consta de tres módulos básicos:

- ✓ El módulo con las librerías que se usan de un modo compartido por el resto de los módulos.

- ✓ El módulo para la gestión de históricos (Likindoy-HTR), y
- ✓ El módulo para la gestión de RTU's (Likindoy-RTU).

3.2.1 Likindoy-HTR

El módulo para la gestión de históricos dispone de 4 niveles de procesamiento de los datos.

1. Recolección de los datos
2. Carga de los datos
3. Generación de gráficas
4. Envío de los datos

3.2.2 Likindoy-RTU

Este módulo de Likindoy tiene las siguientes características:

- ✓ Es capaz de recolectar datos directamente de hardware industrial, como por ejemplo: PLCs Schneider Momentum o sistemas de adquisición de datos ADAM.
- ✓ Lenguaje de comunicaciones es MODBUS sobre TCP/IP.
- ✓ Actúa como un Telemando (hardware de Telecontrol), por lo que es posible descargarse de él los datos por FTP, SFTP o SOCKET.
- ✓ Las versiones más recientes de Likindoy permiten además tomar decisiones y actuar sobre el Hardware gestionado.

- ✓ Soporta cualquier tipo de módulos diseñados por los usuarios.
- ✓ Puede recoger ficheros de datos mediante los protocolos FTP y SFTP o directamente desde un servidor WEB.
- ✓ Servidor MODBUS o mediante el protocolo UDP.

3.2.3 Likindoy-HMI

Significa Human Machine Interface y es el módulo de Likindoy encargado de:

- ✓ Mostrar la información en tiempo real.
- ✓ Esta información es posible conectarla a un mapa de una central depuradora y ver cada elemento cambiando en tiempo real mediante una web.
- ✓ Las posibilidades se encuentran en las limitaciones que el programador encuentre consigo mismo para desarrollar nuevos módulos y herramientas.
- ✓ En la actualidad se está diseñando un sistema basado en C que forme parte de Likindoy y que pueda ser instalado en los sistemas de adquisición de datos ADAM con capacidad para tomar decisiones en tiempo real y almacenar históricos en su memoria interna.

3.3 Tabla Comparativa entre Sistemas SCADA.

En la tabla III.VII se muestra un esquema comparativo de la soluciones OpenSCADA y Likindoy para automatización, soluciones con mayor difusión en el mercado y de uso extendido, bajo licencia pública GPL.

Tabla III.VII. Comparación entre el Sistema OpenSCADA y Sistema Likindoy.

CARACTERÍSTICA	OPENSCADA	LIKINDOY
Desarrolladores	Roman Savochenko, Maxim Lysenko, Yashina Ksenia	Juan Miguel Taboada Godoy
Plataforma hardware Utilizada	PC	PC
Plataforma	Linux (OpenSource)	Linux (OpenSource)
Tipo de Aplicaciones	Modelos dinámicos a tiempo real. Tiempos en ejecución de PLC.	Graficas de Procesos, Visualización de Datos a tiempo real.
Ultima Versión Estable	*OpenSCADA Versión 0.6.4.2 (28.05.2010)*	Likindoy v1.0e (stable) - Release 20080310b
Drivers para Buses	Modbus TCP/IP, SMTP, WEB, Diversos PLC.	Ethernet, Modbus TCP/IP, FTP, SFTP, WEB, UDP, Diversos PLC.
Control de Usuarios	Si.	No.

.

Tabla III.VII. Comparación entre el Sistema OpenSCADA y Sistema Likindoy. (Continuación).

Máxima frecuencia de Muestreo de una Señal	Según PLC.	Según PLC
Lenguaje de Programación	Lenguaje C++	Python
Integración con otras aplicaciones	Java	Java, PHP
Generador de Informes	Modulo Web Visión basado en tecnologías web (XHTML, JavaScript, CSS, AJAX). Archivos en Texto plano.	Graficas con el sistema R. Envío de graficas al correo.
Funcionalidad de Informes	Visualización y control de las zonas controladas mediante datos del modulo VCA Engine (motor de Área de control Visual).	Recolección de datos en archivo de texto plano. Datos transportados a base de datos MySql. Generación de graficas a partir de la base de Datos.

Tabla III.VII. Comparación entre el Sistema OpenSCADA y Sistema Likindoy. (Continuación).

Soporte del Usuario	Web	Web
Licencia	GPLv2	GPLv3
Procesamiento de Datos	Si	Si
Tipo de Arquitectura	Modular	Modular

Fuente: Original.

Por tanto, en el siguiente capítulo se analizará la solución más adecuada para la realización del proyecto, tomando en cuenta lo siguiente:

- ✓ Si el software sigue en desarrollo.
- ✓ Soporte técnico sobre el código y funcionamiento.
- ✓ Facilidad en la programación para ser cambiada a las necesidades del proyecto.
- ✓ Modularidad.
- ✓ Protocolos de Comunicación Industrial soportados.
- ✓ Calidad de Gráficas y facilidad para generarlas.

CAPÍTULO IV

LIKINDOY

En base a las características analizadas en el Capítulo III se montara la solución sobre Likindoy teniendo en cuenta que es el que mejor se ajusta a las necesidades del proyecto. Adicionalmente Likindoy cuenta con ventajas como el tipo de aplicaciones, graficas de proceso, visualización de datos a tiempo real, los drivers para buses que maneja Ethernet, Modbus, TCP/IP, FTP, SFTP, WEB, UDP, Diversos PLC, el lenguaje de programación Python, Generación de informes a través de graficas con el sistema R, envió de graficas por correo, la recolección de datos lo hace en archivo de texto plano, los datos pueden ser transportados a bases de datos MySql, generación de graficas a partir de la base de datos, sigue siendo desarrollado por tanto se puede tener comunicación directamente con el desarrollador.

4.1 Introducción

La creación de Likindoy ha seguido el modelo de desarrollo iterativo y creciente, y se pretende usar mas técnicas de programación extrema, se consiguieron los primeros resultados prácticos en las primeras semanas, desde entonces ha habido una interacción constante con el usuario y por ende una retroalimentación continúa. Ha sido completamente reescrito 2 veces. El modulo Likindoy RTU se esta reescribiendo en la actualidad.

El proyecto se inicio a finales del 2004. Buscando un sistema funcional que generara graficas de manera inmediata, el diseño se baso en programación en R, que tomaba las decisiones a la hora de generar cada grafica. A finales del año 2005 y toda la primera mitad del año 2006 el proyecto se reescribió en Python buscando extender las funcionalidades, el sistema cubría las necesidades iníciales, y se podría crear un sistema SCADA completo dividiendo al sistema en módulos.

Durante los meses de Julio y Agosto del año 2006 el proyecto fue reescrito completamente, se crearon clases, se definieron las configuraciones por defecto, etc...

Ayudados por la subvención de la Consejería de Innovación Tecnológica, se impulsaron las siguientes funcionalidades:

- ✓ Acelerar la adquisición de datos.
- ✓ La modulación del sistema, de forma que se permitiesen datos de entrada y formatos de salida distintos a los inicialmente considerados.

- ✓ Programación de una nueva fuente de datos: Los Momentum de Schneider. Esta modificación supuso también la creación de unas “Unidades Remotas de Telecontrol", que son ordenadores personales (PC) que se comunican con los autómatas y envían los datos registrados en archivos de texto a la unidad central. Esta modificación también ha supuesto la programación del protocolo de comunicación Modbus sobre TCP-IP, que es libre y se está popularizando entre los fabricantes de hardware.

- ✓ Envío de los datos adquiridos por estaciones remotas, por UDP u otros protocolos, a un servidor web, de forma que se pueda ofrecer información relevante al ciudadano en tiempo real (temperaturas, intensidad y dirección del viento).

- ✓ Documentación del proyecto.

En octubre 2007- Enero 2008. Se diseño una RTU programada en C que funcionara directamente dentro de los Advantech ADAM 5510 que tenían una CPU 80188. Cuando el proyecto estaba al 85% aproximadamente el hardware no era lo suficientemente estable para lo que se diseño (se bloqueaba sin dar posibilidad a controlarlo remotamente para reiniciarlo).

En la actualidad, además del mantenimiento y mejoras continuas del software en funcionamiento, se trabaja en varias aéreas:

Reescribiendo Likindoy-RTU, trasladando la experiencia con la RTU en C, con ello se pretendió conseguir una RTU totalmente versátil, que además de adquisición de datos y

servidor de estos vía UDP, sea capaz de envió de alarmas, toma de decisiones y actuación en campo. Es decir, todas las capacidades de una RTU comercial. (PLC)

A la vez ampliamos Likindoy-RTU para su comunicación con unos PLC de la marca Pesyr, por GPRS. Es un proyecto paralelo en el que se busca realizar una versión del servidor UDP de Likindoy que además soporta la entrada de datos de unidades remotas de PESYR que envían sus capturas en paquetes UDP través de GPRS al servidor UDP de Likindoy.

En el futuro se pretende preparar un sistema HMI, es decir, la visualización de los datos en sinópticos y de alarmas en tiempo real, para ello se pretende usar Flash.

4.2 Definición

Likindoy es un SCADA realizado con tecnologías libres, que acerca la automatización y el control de los sistemas industriales a los expertos en informática.

Likindoy es un proyecto que tiene por objeto la creación de un sistema de control y de adquisición de datos (SCADA) a partir de software libre, ordenadores personales (PC) y con uso intensivo de los protocolos y tecnologías popularizados en Internet.

Likindoy intenta ser un sistema simple, fácil de entender por ((hackers)) e ingenieros de control.

Likindoy es una iniciativa de Axaragua, que lo usa como sistema propio de gestión, y ha contado con el incentivo de la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía.

Likindoy se encuentra totalmente inmerso en el concepto de opensource, ya que corre bajo Linux, está programado en Python y su gestor de bases de datos es MySQL.

Este sistema puede ser muy útil a cualquiera que necesite automatizar, controlar y/o representar flujos de datos. Por ejemplo: la temperatura en tu ciudad, el tráfico de red, el consumo de agua en una finca, el trabajo de los motores de una fábrica, aperturas y cierres de las puertas de un edificio, flujo de aviones en un aeropuerto.

Inicialmente el programa realiza representación histórica y representación de valores instantáneos. Aunque el proyecto Likindoy pretende crear un Sistema SCADA completo realizado con tecnologías libres: GNU/Linux, Python, MySQL y R.

4.3 Características

Sus principales características son:

- ✓ Es libre. Por lo que evita costes de licencias, y se puede modificar para conseguir los resultados realmente necesitados.
- ✓ Está escrito en Python. Por lo que la robustez, modularidad y facilidad de mantenimiento del código está garantizada.
- ✓ Usa la base de datos libre MySQL.
- ✓ Usa el sistema R de estadística y generación de graficas, por lo que las graficas resultantes son de una calidad y utilidad desconocida en los sistemas SCADA.
- ✓ Funciona en sistemas con arquitecturas de PC, baratas y universales.
- ✓ Funciona en sistemas basados en Linux. Libre y de gran estabilidad.

Likindoy puede hacer infinidad de cosas, pues tiene una gran flexibilidad y las posibilidades y combinaciones son ilimitadas.

4.4 Objetivos del Likindoy

4.4.1 Graficas de proceso de gran calidad

La calidad, la personalización y la utilidad de las graficas que se ha conseguido son una de las razones de ser de Likindoy y una de las características que lo hacen distinto a cualquier otro SCADA.

4.4.2 SCADA libre

En la actualidad la mayoría de los procesos industriales están controladas por un extenso abanico de software propietario, cuyas desventajas son: el coste de las licencias, su obsolescencia, una gran dependencia tecnológica (no puedo realizar ciertas cosas porque el sistema no me deja), y, lo que es más importante, son sistemas ((cerrados)) poco configurables y adaptables al proceso real (necesitamos una solución determinada que el sistema no da), por lo que terminamos renunciando a conseguir ciertos resultados, o incorporando muchas herramientas distintas para acometer tareas sencillas.

Likindoy, que es software libre, no tiene coste de licencias, y tampoco es fácil que quede obsoleto, pues el usuario siempre puede ir mejorando y ampliando sus capacidades, a lo cual favorece mucho que este desarrollado en Python, un lenguaje interpretado de alto nivel que resulta sencillísimo de leer y mantener.

4.4.3 SCADA en PC, no en PLC

La mayoría de los SCADA’S industriales se basan en sistemas (no estándar), distintos para cada marca: distintas redes, protocolos, PLC propietarios, sistemas HMI distintos, etc.

Detrás de Likindoy reside el concepto de que para los sistemas de control modernos es necesario aplicar tecnologías que en la informática común se entiendan como estándar: Ethernet para comunicación física, Modbus TCP/IP para los protocolos, ftp para la transferencia de ficheros, MySQL para las bases de datos, Flash para el HMI, etc. De esta manera cualquiera que tenga conocimientos suficientes de informática puede montar y mantener un sistema de control complejo.

Por ello es una herramienta ideal para el técnico en informática que quiera adentrarse en el control industrial, y para el técnico de control industrial que o bien tenga una buena base informática, o cuente con apoyo de algún tipo en esa área.

4.5 Etapas de Funcionamiento

Likindoy tiene 4 etapas básicas de funcionamiento, además de las ampliadas por el servidor UDP y otros:

4.5.1 Adquisición de datos: el sistema se conecta a todas las RTU (unidades remotamente) para descargarse los datos de cada una.

4.5.2 Carga de datos: procede a cargar toda la información obtenida en las bases de datos del programa.

4.5.3 Generación de graficas: usa la información disponible en la base de datos para generar las graficas que le sean posibles en función del test de validación asignado a cada grafica.

4.5.4 Envío por email: usando todas las graficas generadas se envía un email a cada usuario de la lista si sus graficas estaban disponible (debemos tener en cuenta que cada usuario puede recibir un grupo distinto de graficas).

4.6 Módulos de likindoy

Likindoy es un programa diseñado para análisis industrial y sistemas de telecontrol. Ha sido diseñado modularmente para permitir a otros programadores ampliar su funcionamiento con facilidad. Likindoy está dividido en varias etapas de funcionamiento aunque internamente se divide en tres módulos básicos:

- ✓ El módulo con las librerías que se usan de un modo compartido por el resto de los módulos,
- ✓ El módulo para la gestión de históricos (Likindoy-HTR) y
- ✓ El módulo para la gestión de RTU's (Likindoy-RTU).

4.6.1 Likindoy-HTR

El módulo para la gestión de históricos dispone de 4 niveles de procesamiento de los datos.

4.6.1.1 Recolección de los datos:

Donde el programa usa diferente módulos para conseguir la información de fuentes externas. (FTP, SFTP, SOCKECT, /PROC, MODBUS, WEB, UDP).

4.6.1.2 Carga de los datos:

Recoge los datos descargados y los transfiere de modo homogéneo a una base de datos SQL (opcionalmente podría ser cualquier otro sistema de salida, incluso un programa que leerá los datos en código Morse por el altavoz).

4.6.1.3 Generación de gráficas:

Usa la información almacenada en la base de datos para generar gráficas. El motor para pintar los datos está bajo la librería RPY (librería Python para comunicación con el lenguaje estadístico R) y en la actualidad existe una propuesta para el diseño de un nuevo módulo para pintar los datos a través de Python sin usar el motor de R.

4.6.1.4 Envío de los datos:

Envía los gráficos generalmente por email, pero soporta también FTP, SFTP y SOCKET (los usuarios pueden diseñar nuevos módulos para enviar los datos mediante sus propios protocolos).

En la figura IV.5 se presenta un ejemplo de Likindoy-HTR que representa la dirección e intensidad del viento, la humedad relativa y la temperatura de una estación meteorológica.

Figura IV.5. Grafica Likindoy-HTR

Esta gráfica representa el caudal de agua suministrada, nivel de depósito, nivel de cloro y estado de avería de los equipos (representado por una línea horizontal) de una planta de producción de agua potable, en un intervalo de tiempo 24 horas. Los puntos rojos representan los caudales medios diarios de los últimos 24 días. Los triángulos verdes y rojos representan picos de caudal (la avería de la válvula no es real, es debido a un fallo de conexiones en campo).

Likindoy puede enviar las gráficas junto con muchas otras todos los días por email.

4.6.2 Likindoy-RTU

Es el módulo de Likindoy capaz de recolectar datos directamente de hardware industrial, como por ejemplo: PLCs Schneider Momentum o sistemas de adquisición de datos ADAM 5000 TCP, aunque el lenguaje de comunicaciones es MODBUS sobre TCP/IP y esto permite comunicarse casi con cualquier PLC industrial. Este módulo además de la adquisición de los datos actúa como un Telemando (hardware de Telecontrol), por lo que es posible descargarse de él los datos por FTP, SFTP o SOCKET. Likindoy-HTR generalmente pregunta al Telemando (que usa Likindoy-RTU) por los datos que necesita, Likindoy-RTU permite entonces a Likindoy-HTR que pueda descargarse esos datos y cuando ha terminado la borra para liberar memoria en la RTU. Las versiones más recientes de Likindoy permiten además tomar decisiones y actuar sobre el Hardware gestionado.

Likindoy ha sido diseñado para soportar cualquier tipo de módulos diseñados por los usuarios, gracias a ello, en la actualidad Likindoy-RTU puede registrar datos del tráfico de la red obtenido del sistema de ficheros /proc, puede recoger ficheros de datos

mediante los protocolos FTP y SFTP o directamente desde un servidor WEB, servidor MODBUS o mediante el protocolo UDP.

4.6.3 Likindoy-HMI

Significa Human Machine Interface y es el módulo de Likindoy encargado de mostrar la información en tiempo real. Es un conjunto de funciones y comportamientos que habilitan a Likindoy para que pueda mostrar en tiempo real la información que está procesando. Esta información es posible conectarla a un mapa de una central depuradora de aguas y ver cada elemento cambiando en tiempo real mediante una web desde casa. Las posibilidades se encuentran en las limitaciones que el programador encuentre consigo mismo para desarrollar nuevos módulos y herramientas.

En la figura IV.6 se muestra un ejemplo de una interface Hombre Maquina

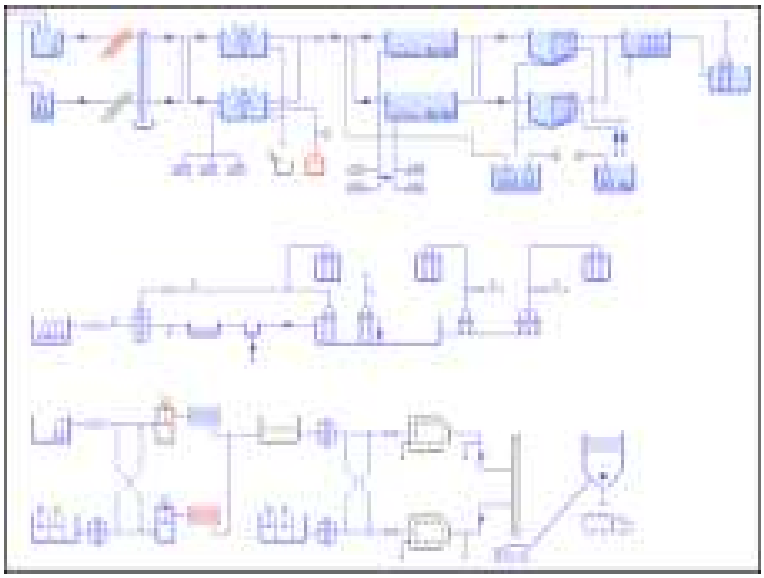


Figura IV.6. Sistema Likindoy - HMI

4.7 Detalles del sistema

4.7.1 Análisis funcional

Desde el punto de vista funcional, Likindoy supone un gran avance en el área de informática industrial, debido principalmente a dos factores muy importantes:

- ✓ Acerca los sistemas industriales a los Ingenieros Informáticos y
- ✓ Los costes de implantación son muy reducidos.

4.7.1.1 Alcance funcional

Likindoy puede trabajar en un entorno mixto donde se atiende a la vez a PLCs, RTU's de Likindoy, PCs conectados a la red, clientes UDP y unidades remotas, además la misma máquina puede hacer la generación de datos y publicación de los mismos (por web, por email, por UDP u otros protocolos).

Likindoy puede cubrir con éxito el seguimiento de históricos, esto significa que puede generar graficas o compartir datos en tiempo real puesto que pasado 1 segundo del momento actual ya se considera un histórico para Likindoy y esto se puede extender en el tiempo hasta centenares de años si es necesario.

Obviamente la obtención, carga de datos y generación en la base de datos debería ser más rápida de dicho segundo, lo cual no es viable en la generación de graficas aunque sí lo es en la compartición de datos por UDP, WEB u otros.

Además el sistema de RTU permite tomar decisiones con tiempos de retraso no superiores a 1 segundo (dependiendo del retraso de la red y del número de señales procesadas).

Likindoy puede ser usado en cualquier entorno industrial que requiera un seguimiento del estado de los sistemas de un modo parcial, es decir, que ocurrió en las últimas 4 horas, que ocurrió ayer, etc. y usar esta información para realizar previsiones de mantenimiento y ordenar el trabajo de los operarios en consecuencia con la información obtenida. Igualmente permite conocer que parámetros ajustar para optimizar los medios de producción y además ayuda a conocer de un vistazo de los efectos laterales de ciertas acciones que no veríamos en el proceso en sí, es decir, si llueve el humo producido por la planta se condensa antes en la salida produciendo una lluvia interna que provoca un aumento general de la presión en la etapa de salida.

Likindoy puede ser usado en sistemas de servidores para controlar información de la red de tal modo que sea posible conocer cuando un sistema está consumiendo más Internet de lo que debería y es posible atender a un seguimiento para encontrar las causas, situación imposible de descubrir si el evento ocurre ocasionalmente y que quedaría patente en una grafica.

Incluso puede ser ampliado, cualquier programador con unos conocimientos básicos de Python puede ampliar Likindoy con nuevos módulos para atender a otros sistemas usando otros protocolos.

Además Likindoy-RTU permite un control en tiempo real para acciones y reacciones que requieren tomar decisiones en el instante y de un modo automático (sin la decisión de un operador).

4.7.1.2 Casos de Éxito

Gracias a Axaragua, Likindoy se ha ido completando con un amplio rango de casos de éxito que confirman la funcionalidad y estabilidad del sistema.

Gráficas diarias: La generación de graficas diarias de los sistemas de Axaragua nos ayuda a prevenir situaciones de riesgo, roturas de motores y plantear los mantenimientos diarios. Tanto gerente, como encargados de planta, operarios e incluso clientes reciben las graficas de Likindoy, de tal modo que cada cual actúa en consecuencia con sus responsabilidades.

Likinweather: Nos permite mostrar información del estado actual de la meteorología en tiempo real.

Análisis de red: Igualmente tanto de Axaragua como de Centrologic se generan graficas del flujo de la red en ellas podemos ver si una conexión está saturada por exceso de uso, de donde viene la carga que se produce, como se comporta durante la noche, etc. Una información muy útil para administradores de grandes redes.

Graficas en la web: Además Likindoy nos permite mostrar graficas en la web para que los visitantes puedan conocer como varió el estado ayer o consultar todo el histórico del estado de los procesos hasta la fecha actual.

4.8 Sistema estadístico R

R es un entorno de software libre para el cálculo estadístico y gráficos. Se compila y se ejecuta en una amplia variedad de plataformas UNIX, Windows y MacOS. Es un lenguaje y un entorno para computación y gráficos estadísticos, además es un proyecto

GNU, que fue desarrollado en los Laboratorios Bell (Lucent Technologies) de John Chambers y colegas.

R ofrece una gran variedad de estadísticas (modelos lineales y no lineales, clásicas pruebas estadísticas, análisis de series temporales, clasificación, agrupación, etc.) y técnicas gráficas, además es muy extensible.

Uno de los puntos fuertes de R es la facilidad con la que se puede diseñar y producir publicaciones de buena calidad, incluyendo símbolos matemáticos y las formulas que sean necesarias, aunque se debe tener cuidado con los valores predeterminados para las opciones de diseño en los gráficos, pero el usuario mantiene el control completamente.

R está disponible bajo los términos de la Fundación de Software Libre, y está disponible en forma de código fuente.

4.8.1 Características:

- ✓ R es un conjunto integrado de servicios de software para la manipulación de datos, cálculo y representación gráfica.
- ✓ Tiene un eficaz manejo y almacenamiento de datos.
- ✓ Un conjunto de operadores para los cálculos con arreglos, en particular, matrices.
- ✓ Una gran colección de herramientas de análisis de datos intermedios.
- ✓ Facilidades gráficas para el análisis y visualización de datos ya sea en pantalla o en papel, y

- ✓ Un bien desarrollado, sencillo, pero eficaz lenguaje de programación, que incluye condicionales, bucles, funciones recursivas se definen nuevos usuarios y funciones con facilidades de entrada y salida.

- ✓ Las instrucciones (cálculos, formatos de gráficos, etc), se pueden almacenar para realizar futuras operaciones en archivos denominados “paquetes”, que permitirán realizar similares operaciones sin necesidad de volver a escribir nuevamente las formulas o funciones. En este aspecto cabe destacar que los usuarios y/o programadores, que se han tomado el trabajo de desarrollar estos paquetes, los comparten libremente dentro del proyecto. Es así que en la página del proyecto se pueden conseguir cientos de paquetes con sus respectivos instructivos para su uso.

- ✓ El término "medio ambiente" se refiere a la característica de ser un sistema planificado y coherente, en lugar de una acumulación gradual de herramientas, muy inflexible, como frecuentemente sucede con otro software de análisis de datos.

- ✓ R está diseñado entorno a un autentico lenguaje de programación, y permite a los usuarios añadir funcionalidades para definir nuevas funciones y facilita a los usuarios a seguir las decisiones tomadas por el algoritmo.

- ✓ Para tareas computacionalmente intensivas, se puede enlazar condigo C, C + + y Fortran, además se puede llamar en tiempo de ejecución.

- ✓ Los usuarios avanzados pueden escribir código C para manipular objetos R directamente.

- ✓ R puede integrarse con distintas bases de datos y existen librerías que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python.

- ✓ R también puede usarse como herramienta de cálculo numérico, campo en el que puede ser tan eficaz como otras herramientas específicas tales como GNU Octave y su versión comercial, MATLAB.[4] Se ha desarrollado un interfaz, RWeka[5] para interactuar con Weka que permite leer y escribir ficheros en el formato *arff* y enriquecer R con los algoritmos de minería de datos de dicha plataforma.

- ✓ Muchos usuarios piensan de R como un sistema de estadísticas. Preferimos pensar en él de un entorno en el que las técnicas estadísticas se aplican.

- ✓ R tiene su documentación en un formato propio (LaTeX), que se utiliza para suministrar amplia documentación, tanto en línea, en varios formatos y en papel.

4.8.2 Extensiones y Paquetes

R forma parte de un proyecto colaborativo y abierto. Sus usuarios pueden publicar paquetes que extienden su configuración básica. Existe un repositorio oficial de paquetes cuyo número superó en otoño de 2009 la cifra de los 2000.

Dado el enorme número de nuevos paquetes, éstos se han organizado en vistas (o temas), que permiten agruparlos según su naturaleza y función. Por ejemplo, hay grupos de paquetes relacionados con estadística bayesiana, econometría, series temporales, etc.

Para facilitar el desarrollo de nuevos paquetes, se ha puesto al servicio de la comunidad una forja de desarrollo que facilita las tareas relativas a dicho proceso.

CAPÍTULO V

DESCRIPCIÓN DEL EQUIPAMIENTO

5.1 Sistema de Monitoreo de Energía

Un sistema de monitoreo de energía es una red de medidores o analizadores de redes interconectados. Esta interconexión también puede incluir un servidor central.

El sistema puede:

- ✓ Concentrar la información de equipos de medición y tenerla disponible en interfaces de fácil manejo como los navegadores Web.
- ✓ Generar la información de costos de energía, demanda, factor de potencia, consumos, haciendo uso de mediciones en tiempo real, con los algoritmos que indica la Secretaría de Energía.

- ✓ Generar reportes de eventos de Calidad de la Energía (gráficos, tendencias, espectros, formas de onda).
- ✓ Enviar alarmas de comportamientos anormales en la red eléctrica, interrupciones en suministros. Estas son generadas directamente en el monitor de circuitos y enviados a internet, pueden ser recibidos vía SMS o e-mail.

5.2 Ventajas de la medición y análisis

- ✓ Oportuna detección de causas de falla de equipos.
- ✓ Verificación de facturas.
- ✓ Son imprescindibles en los programas de ahorro de energía.
- ✓ Permiten implementar mejores estrategias de mantenimiento.
- ✓ Nos posibilitan el asignar costos a áreas específicas.
- ✓ Buena información.
- ✓ Buenas decisiones.

5.3 Schneider Electric

Power Measurement es ahora parte de Schneider Electric y su marca PowerLogic de soluciones de gestión de energía.

5.3.1 Oferta de Schneider Electric en monitoreo y control de la energía

Power Logic es el sistema de medición más poderoso del mercado. Su complemento perfecto es el software de análisis y despliegue de información SMS. Los medidores ION son los más precisos para facturación, el software ION Enterprise es muy versátil y permite el desarrollo de aplicaciones personalizadas profesionales. La oferta en medición de Schneider electric es la más completa del mercado porque tiene medidores para todas las aplicaciones que cubren con todas las necesidades de nuestros clientes.

La versatilidad de los equipos les permite comunicarse con dispositivos de casi cualquier marca y están diseñados para trabajar en redes de:

- ✓ Agua
- ✓ Gas.
- ✓ Eléctricas, etc.

La empresa también integra otros insumos con los mejores equipos de medición del mercado:

- ✓ Usando protocolos abiertos de comunicación.
- ✓ Con medición de pulsos.
- ✓ Entradas analógicas.
- ✓ Contactos Secos

Los medidores pueden integrarse fácilmente a un sistema de administración de energía o sistema SCADA a fin de ofrecer una visualización remota de todos los parámetros

medidos en una estación de trabajo PC así como funciones de configuración remota y control manual.

5.4 Medidor ION 7550 / 7650

Los medidores ION 7550 e ION 7650 son utilizados en puntos clave de distribución y cargas sensitivas, ofrecen una funcionalidad inigualable, que incluye análisis avanzados de la calidad de energía asociados con la precisión de facturación, opciones múltiples de comunicación, compatibilidad con la web y capacidades de control.

Los medidores incluyen una extensa selección de pantallas de datos pre-configurados y medidas que usted puede utilizar inmediatamente como predeterminadas o puede adaptarlos a sus necesidades específicas.

Para una solución de administración de energía empresarial, los medidores pueden integrarse con el software ION EEM, ION Enterprise, o bien otro software de administración de energía y sistemas SCADA a través de canales múltiples de comunicación y protocolos.

La tecnología patentada ION también le permite personalizar las funciones de medición y análisis en su estación de trabajo sin necesidad de conexiones. Sólo tiene que unir gráficamente unos iconos de arrastrar y soltar o bien seleccionar unos parámetros predeterminados y ya puede empezar.

5.4.1 Mediciones

- ✓ Excede la Clase 0.2 de precisión de facturación.
- ✓ Voltaje de 3 fases Instantáneo, corriente, frecuencia y factor de potencia.

- ✓ Energía: bidireccional, absoluta y neta, tiempo de uso, pérdida de compensación.
- ✓ Demanda: Sliding window (bloque rotatorio), prevista, y térmica.
- ✓ Armónicas: THD distorsión armónica individual y total hasta la armónica 63 (511 en software)
- ✓ Detección de transitorios: 17us a 60 Hz (20us a 50 Hz) y almacenamiento de sag/swell

5.4.2 Comunicaciones Compatibles con Internet

- ✓ WebMeter, MeterM@il® que permite la distribución de información, medida y alarmas por medio de Internet.
- ✓ Como una opción que puede ser incorporada tiene el Módem integrado con ModemGate que permite el acceso por medio de módem para otros 31 dispositivos.
- ✓ Puerto Ethernet Base10-T ó Base 10 –FL con EtherGate, que permite transferencia de datos directa desde Ethernet a puerto RS-485.
- ✓ Dos puertos RS-485, uno conmutable a RS-232.
- ✓ Un puerto óptico en el panel frontal ANSI, Tipo 2.
- ✓ Soporte para los protocolos Modbus RTU/TCP y DNP 3.0.
- ✓ Soporte para Modbus Master.

5.4.3 Almacenamiento de Datos en Memoria Interna

El almacenamiento puede darse en forma programada o derivada de eventos, además almacena secuencia de eventos, mínimas / máximas de corriente o voltaje, formas de onda, fallos y cargas de transitorios.

5.4.4 Entradas y Salidas

El formato estándar incluye 8 entradas digitales, 3 salidas de regulador Forma C (electromecánica) para funciones de control, y 4 salidas digitales Forma A (estado sólido) para funciones de pulsos. También disponible con 8 entradas digitales adicionales, 4 salidas análogas, y/ó 4 entradas análogas.

Los medidores muestran la fiabilidad del sistema mediante nueves (por ej. 99.99% del tiempo productivo)

Figura V.7. Fiabiladad del sistema

Tendencias de uso de KWh directamente en el panel frontal

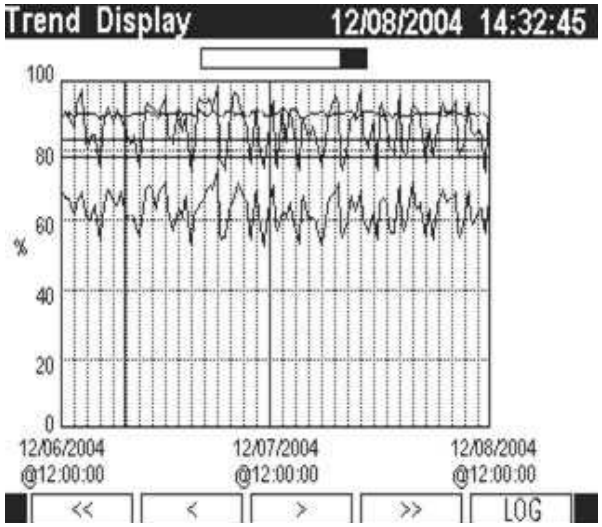


Figura V.8. Panel Frontal

5.4.5 Medición

5.4.5.1 Energía

Los medidores son completamente bidireccionales, y monitorean energía en cuatro cuadrantes. Los medidores ofrecen los parámetros de energía activos, reactivos y aparentes, y pueden integrar cualquier parámetro de potencia para ofrecer medidas como:

- ✓ kWh (producidos y recibidos)
- ✓ kWh, kVARh, kVAh netos (producidos menos recibidos)
- ✓ kWh, kVARh, kVAh total (producidos + recibidos)
- ✓ kVARh, kVAh producidos y recibidos
- ✓ Voltios-hora y Amperios-hora

Puede integrar cualquier medición instantánea, además los registros de energía pueden ser cargados automáticamente en horarios programados.

5.4.6 Medición de la Calidad de Energía

5.4.6.1 Control de Cumplimiento

- ✓ Norma de monitoreo EN 50160
- ✓ Norma IEC 61000-4-7 de armónicos e inter armónicos **
- ✓ Norma IEC 61000-4-15 de señalización de fluctuaciones
- ✓ Normas IEEE 519 y IEEE 1159

5.4.6.2 Registro de las Formas de Onda

- ✓ Los medidores pueden captar simultáneamente todos los canales de voltaje y corriente.
- ✓ Captura de alteración de subciclos
- ✓ El número máximo de ciclos por captura de forma de onda contigua es 214,000 (basada en 16 muestras / ciclo x 96 ciclos y la más amplia capacidad de memoria del medidor)
- ✓ Hasta 512 muestras / ciclo estándar, 1024 muestras / ciclo opcional con el medidor ION 7650
- ✓ Hasta 256 muestras / ciclo con el medidor ION 7550
- ✓ Rango dinámico: Entradas de voltaje - 16 bits efectivos, Entradas de corriente -

- ✓ 19 bits efectivos

5.4.7 Medición del Tiempo Productivo

5.4.7.1 Utilizando Nueves

La infraestructura actual de alimentación eléctrica puede suministrar electricidad con una fiabilidad del 99.9% (3 nueves ó 8.8 horas de tiempo de parada al año). No obstante, cualquier interrupción es inaceptable para los negocios de la economía digital, los cuales pueden requerir hasta un 99.9999999% (9 nueves ó 2 ciclos de tiempo de parada al año) para que su modelo empresarial opere eficazmente. Con los medidores ION 7550 e ION 7650 puede medir el número de nueves correspondiente a la fiabilidad.

5.4.7.2 Detección Fuera de Límite

Detecta, registra y genera informes sobre las características específicas de los desequilibrios de voltaje y corriente y las variaciones en las pérdidas y el factor de potencia, las sobretensiones o subtensiones, etc.

5.4.8 Comunicaciones

5.4.8.1 Acceso multipuerto, multiprotocolo

Comunicación simultánea con hasta 4 puertos proporciona una compartición segura de datos con una variedad de sistemas de administración de energía usando una selección de opciones de normas y protocolos de comunicación.

5.4.8.2 Puerto RS-232/RS-485

- ✓ Seleccionables entre RS-232 y RS-485

- ✓ Protocolos: ION, DNP 3.0, Modbus RTU, GPS, EtherGate, ModemGate o Modbus Máster
- ✓ Velocidad en baudios: 300bps a 115,200bps.

5.4.8.3 Puerto RS-485

- ✓ Protocolos: ION, DNP 3.0, Modbus RTU, GPS, EtherGate, ModemGate, o Modbus Master
- ✓ Velocidad en baudios: 300bps a 57,600bps

5.4.8.4 Puerto de Datos Infrarrojos

Cuenta con un puerto Óptico ANSI Tipo 2 puede descargar información en tiempo real a una computadora personal.

- ✓ Protocolos: ION, Modbus RTU, DNP 3.0
- ✓ Velocidad en baudios: 1200 bps hasta 19,200 bps.

5.4.8.5 Módem Interno

Módem interno de teléfono disponible, que ofrece una conexión rápida y ModemGate, un portal de red, permite compartir el módem interno del medidor a través de los puertos disponibles de serie con hasta 31 dispositivos adicionales.

- ✓ Protocolos: ION, Modbus RTU, DNP 3.0
- ✓ Velocidad en baudios: Hasta 33.6 Kbps

5.4.8.6 Puerto Ethernet

El puerto es opcional, puede ser Base 10-T ó Base 10-FL, ofrece acceso directo a través de un LAN/WAN de Ethernet e incorpora EtherGate, un sistema que permite transferir datos directamente entre una red Ethernet a un máximo de 62 dispositivos a través de los dos puertos de serie del medidor.

- ✓ Protocolos: TCP/IP, ION, Modbus TCP, DNP 3.0, Telnet, NTP, DNS y SMTP
- ✓ Velocidad en baudios: Hasta 10 Mbps.

5.4.9 Interoperatividad

Tiene que ver con la habilidad de comunicaciones simultáneas mediante protocolos múltiples, le permiten utilizar funciones avanzadas de cualquier medidor para expandir una red existente de Modbus, DNP o ION. Los registros y valores en tiempo real también están disponibles mediante Modbus.

5.4.10 Soporte para Software UTS

Los medidores son totalmente compatibles con plataformas de software UTS incluyendo MV-90®, MVP, MVRS, MVLT y MVCOMM, y ofrecen una conexión directa entre Ethernet a MV-90.

5.4.11 Firmware basado en Módulos Flash

Se puede realizar actualizaciones mediante comunicaciones, sin retirar la unidad de su emplazamiento.

Tabla V.VIII. Especificación de Medidas.

Parámetro	Precisión ± (%Lectura)
	1 segundo
Voltaje (l-l) (l-n)	0.1%
Frecuencia	±0.005Hz
Corriente (I1, I2, I3)	0.1%
Corriente (I4, I5)	0.4%
kW, kVAR, kVA	clase 0.2*
kWh, kVARh, kVAh	clase 0.2*
Demandas KW, KVA	clase 0.2*
Factor de Potencia (a Unidad PF)	0.2%
Armónica (hasta la 40)#	IEC 61000-4-7
Armónica (hasta la 63)	1% Escala Total
Factor K	5% Escala Total
Factor Cresta	1% Escala Total
Componentes simétricos#	Voltaje:0.2% FS**, Corriente 0.4% FS**

†Referirse al manual del usuario para rangos validos de medidas *Referirse a la sección de Normas en la página 7

Solo en medidor ION 7650 ** FS=Escala Total

La resolución de la pantalla cumple o excede con la precisión.

5.5 Medidor ION 7300/7330/7350

Los medidores ION 7300 son un sustituto ideal de los medidores análogos dado que ofrecen una multitud de mediciones de la potencia y energía, E/S análogas y digitales, puertos de comunicación y protocolos estándar de la industria. El medidor ION 7330 también incluye una función de almacenamiento de datos, correos electrónicos de información cargada, y un módem opcional. El medidor ION 7350 presenta un análisis de la calidad de la energía más sofisticado, alarmas y la función de llamadas de alarma.

Los medidores de la serie Ion 7300 pueden incorporar diversas combinaciones de E/S digitales y análogas para mediciones universales. Estos medidores pueden sustituir los PLC y RTU (transductores para el monitoreo de la presión, la temperatura y la potencia) así como los transductores de potencia tradicionales. Los medidores ION 7330 e ION

7350 aceptan impulsos de entrada de equipos de gas, agua y vapor así como de otros equipos de medición y pueden convertir dichos impulsos en valores de consumo real.

5.5.1 Mediciones

Los productos de la serie ION 7300 ofrecen mediciones completas, bidireccionales, en los 4 cuadrantes, de alta precisión y homologadas para ingresos. Sustituyen los medidores de energía discretos, los medidores de demanda y los indicadores de impulsos y realizan una amplia gama de funciones de medición e instrumentación.

- ✓ Energía: bidireccional, absoluta y neta
- ✓ Demanda: Sliding window (bloque rotatorio), prevista, y térmica
- ✓ Armónicas: distorsión armónica individual y total hasta la armónica 15 ó 31
- ✓ Funciones lógicas y matemáticas avanzadas

Los medidores ION 7330 e ION 7350 pueden utilizar operadores lógicos y Puntos de Ajuste (Setpoints) para ajustar alarmas, implementar protección de respaldo avanzada para equipos, y para definir algoritmos de control básico para el control del condensador y de la demanda. 12 Puntos de Ajuste (Setpoints) son configurables para operaciones de 1 segundo. Cada Punto de Ajuste (Setpoint) puede activarse en las condiciones superiores o inferiores que especifique.

Utilice los Puntos de Ajuste (Setpoints) para activar:

- ✓ La entrada de datos
- ✓ Las salidas digitales

- ✓ Las funciones de borrado y re inicialización
- ✓ Las salidas de impulsos
- ✓ Rellamada (ION 7350)

5.5.2 Comunicaciones

- ✓ Dos puertos RS-485
- ✓ Módem opcional integrado con ModemGate™ que permite acceder con el módem a 31 dispositivos adicionales
- ✓ Puerto opcional Ethernet con EtherGate™ para la transmisión directa de datos de Ethernet a RS-485 y a los 31 dispositivos adicionales
- ✓ Puerto estándar de datos infrarrojos
- ✓ Soporte para el protocolo ModbusTM RTU, Modbus TCP, DNP 3.0 y PROFIBUS DP
- ✓ Función de Rellamada con respuesta rápida de alarma
- ✓ El servidor web y Meter M@il® permiten distribuir por Internet los datos medidos y las alarmas

5.5.2.1 Puertos seriales

El medidor ION 7300 está equipado con un puerto único RS-485, mientras que los medidores ION 7330 e ION 7350 pueden tener dos puertos RS -485, dependiendo de las opciones de comunicaciones seleccionadas.

- ✓ Ópticamente aislados
- ✓ Velocidades en baudios de hasta 19.200 bps.
- ✓ Compatible con software de control de la potencia que ofrece soporte para Modbus RTU o ION
- ✓ El medidor ION 7330 y el medidor ION 7350 también ofrecen soporte para DNP 3.0

5.5.2.2 Puerto de datos infrarrojos

Todos los modelos de la serie ION 7300 ofrecen un puerto óptico en el panel frontal.

- ✓ Compatible con un acoplador de comunicación magneto óptico ANSI C12.13 Tipo II y puede funcionar a velocidades en baudios de hasta 19.200 bps.
- ✓ Para impulsos o comunicación de energía de infrarrojos con nuestro propio software de control de la potencia

5.5.2.3 Puerto Ethernet (opcional)

Todos los medidores en la serie ION 7300 pueden ser ordenados con un puerto opcional 10Base-T para el acceso directo a información de medición por medio de una red Ethernet LAN/WAN.

- ✓ Protocolos: ION, Modbus TCP
- ✓ Velocidad de transmisión de datos: 10 Mbps
- ✓ Servicios de diagnóstico Ping y Telnet

- ✓ EtherGate™ permite al ION 7330 y al ION 7350 trabajar a modo de acceso y permitir la transferencia directa de datos entre una red Ethernet y hasta 31 dispositivos RS-485

El puerto COM2 del medidor funciona como un puerto EtherGate dedicado (RS-485 Master) en los medidores ION 7330 e ION 7350 con la opción de Ethernet.

5.5.2.4 Puerto PROFIBUS (opcional)

El soporte del protocolo estándar PROFIBUS DP mediante un conector hembra sub-D de 9 pins se ofrece exclusivamente con el ION 7300.

5.5.2.4 Módem interno

El ION 7330 y el ION 7350 disponen de un módem interno opcional que ayuda a reducir los costos y mejorar la fiabilidad en sustitución de los módems externos y los convertidores RS-485 a RS-232.

- ✓ Velocidad en baudios de 300 bps a 33.600 bps.
- ✓ La función “ModemGate” permite a la estación maestra remota acceder al medidor y a 31 dispositivos adicionales conectados al bucle RS-485 mediante un único módem interno

El puerto COM1 del medidor funciona como un puerto ModemGate dedicado (RS-485

Master) en los medidores ION 7330 y ION 7350 con la opción de módem interno

- ✓ Compatible con el software de monitoreo de energía que ofrece soporte para Modbus TRU, ION o DNP 3.0

- ✓ RJ-11 o conector de cable cautivo
- ✓ El medidor ION 7350 dispone de una función de Rellamada para una respuesta rápida en caso de alarma.

5.5.2.5 Interoperatividad

Los medidores ION 7330 e ION 7350 pueden establecer comunicaciones simultáneas mediante protocolos múltiples y, por lo tanto, se pueden utilizar para ampliar una red Modbus, DNP o ION Enterprise existente. También pueden obtenerse registros y valores en tiempo real mediante Modbus. Además, los medidores están totalmente soportados por UTS MV-90® mediante los puertos seriales y Ethernet.

Los medidores de la serie ION 7300 soportan demanda sliding window (bloque rotatorio), térmica y prevista. Los medidores calculan la demanda en cualquier valor instantáneo y registran la demanda pico (máxima) y mínima. Los registros de la demanda pico pueden reajustarse manualmente (protegidos por contraseña) desde el panel frontal u otras vías de comunicaciones. Configuración preestablecida:

- ✓ Demanda Kw y mín/máx.
- ✓ Demanda kVAR y mín/máx.
- ✓ Demanda kVA y mín/máx.
- ✓ Demanda de amperios y mín/máx.
- ✓ Demanda de voltios y mín/máx.
- ✓ Demanda de cualquier medición instantánea

5.5.3 Entradas y registros

El ION 7330 y el ION 7350 proporcionan registros de datos y eventos. La memoria no volátil garantiza la preservación de información valiosa entre intervalos cuando los registros se recuperan mediante la función de comunicaciones.

5.5.3.1 Registros históricos

Puede registrase cualquier combinación de mediciones en intervalos programados o mediante Puntos de Ajuste (Setpoints), condiciones lógicas o bien manualmente.

Los medidores de la serie ION 7300 pueden ser configurados para hasta 30 días de registro a intervalos de 15 minutos.

El medidor ION 7350 ofrece un máximo de 6 registros de datos, cada uno registra hasta 16 parámetros definidos por el usuario concurrentemente, dando un total de 96 parámetros

El medidor ION 7330 ofrece un máximo de 2 registros de datos, cada uno registra hasta 16 parámetros definidos por el usuario concurrentemente, dando un total de 32 parámetros.

5.5.3.2 Registro mín. / máx.

- ✓ Realiza un registro mínimo/máximo de cualquier parámetro en cualquier intervalo de tiempo (por ejemplo, diario o mensual)
- ✓ Registro fácil de otros valores que coincidan con el nuevo mínimo o máximo

- ✓ Valores preestablecidos: Mínimo y máximo para todos los parámetros de potencia básicos.
- ✓ Voltaje (1-1/I-n) por fase, corriente por fase, Kw, kVAR, kVA, factores de potencia, frecuencia y demanda de Sliding window de Kw, kVAR y kVA

5.5.3.3 Registro y alarma de eventos

La configuración de prioridad de eventos le permite definir las condiciones de alarma.

- ✓ Secuencia de eventos con fecha y hora ± 10 ms de precisión
- ✓ Registro con hora de todos los cambios de configuración, Puntos de Ajuste (Setpoints) y de eventos mín /máx.

5.6 SEL

Los relés digitales de protección son diseñados para responder a las fallas del sistema, tales como cables de electricidad caídos causados por accidentes o inclemencias del tiempo, en milisegundos. Los Equipos marca SEL sirven a miles de empresas de servicios públicos y, a su vez, millones de clientes, contribuyendo a realizar operaciones más seguras y reducir al mínimo las interrupciones y daños causados por estas fallas.

Entre los muchos productos fabricados por SEL y los cuales comercializan se pueden nombrar las siguientes:

Relés Protectores:

- ✓ Protección de Distribución: Relé de protección del alimentador, Intensidad direccional y Reconectador, protección y control de Disyuntor, Control de regulador de voltaje y control de Reconector entre otros.

- ✓ Protección de Generador y Motor: Relé de Generador de Motor, de protección del motor entre otros

- ✓ Protección Energía Industrial: Relé del motor, del relevo de protección del motor, Relevo de generador, Relevo corriente diferencial entre otros

- ✓ Protección de Transformador, Bus, Interruptor y condensador: Relé de Protección de Transformador del Relevo, Diferencial de corriente y Relevo de voltaje, Corriente diferencial y Sobre corriente

- ✓ Protección de Transmisión: Relé Línea de alta velocidad protección, automatización y control del sistema, Protección y automatización diferenciadas de la corriente del sistema, Relés avanzado de la distancia con recloser, de fase y distancia a tierra y sobre intensidad de corriente direccional entre otros.

Otros de los productos que comercializa SEL, son los indicadores de falla y Sensores, disponibles en varios modelos con múltiples opciones para cumplir los requerimientos de los clientes. La línea de productos incluye modelos para líneas aéreas, cables subterráneos, transformadores de montaje en pedestal, tableros de montaje en pedestal y bóvedas subterráneas. Los indicadores de fallas SEL LINAM ® y SEL AutoRANGERTM son necesarios, en muchos de los casos, para indicarle a las cuadrillas de las compañías eléctricas, a partir de qué punto deben abrir los seccionadores para buscar fallas en redes de distribución, con lo cual se restablece la

energía a algunos de los usuarios y sólo se deja sin ella a los usuarios estrictamente necesarios hasta encontrar la falla.

5.6.1 Relé SEL-751A

El SEL-751A alimentador de relés de protección es la solución adecuada para uso industrial y la protección del alimentador de utilidad, con E/S flexible y opciones fáciles crecientes. Proporciona una excepcional combinación de protección, vigilancia, control y comunicación en un paquete industrial.

Proporciona una mejor protección de alimentación completa, con sobre corriente, sobretensión, baja tensión, y los elementos de frecuencia. Actualiza fácilmente la protección sin cortar o perforar recortes existentes con un factor de forma pequeño y múltiples adaptadores de montaje. Integra rápidamente en serie y basados en la comunicación de Ethernet con la norma IEC 61850, Mirrored Bits ®, DeviceNet ™, y otros protocolos.

5.6.1.1 Características:

- ✓ **Protección completa del alimentador.**

Maximizar la flexibilidad del esquema de control en el tiempo e instantáneo de sobre corriente, sobretensión, baja tensión, y los elementos de la frecuencia con la protección del interruptor de emergencia con un interruptor de tres polos.

- ✓ **Protección de características opcionales.**

El SEL-751A con una de las opciones de voltaje y de entrada y subfrecuencia para proporcionar la tasa de cambio de frecuencia y subfrecuencia, detección sobre funciones

de voltaje, realiza comprobaciones de sincronismo, control de estado de DC de la batería, detección de arco eléctrico, medición de la energía y la demanda.

- ✓ **Los controles convenientes.**

Los cuatro botones programables en el panel frontal para el control rápido y personalizado.

- ✓ **Fácil de Comunicaciones.**

Elección entre Ethernet o comunicaciones sucesivas y varios protocolos, incluyendo Mirrored Bits® y IEC 61850. Elija MODBUS ® TCP o múltiples sesiones Modbus Sucesivas, DNP3 LAN / WAN, o configuración personalizada en serie de DNP3 de las aplicaciones.

- ✓ **Control de Ecuaciones SELogic® Ampliado**.

Usa las matemáticas y la lógica de las combinaciones de valores analógicos y digitales para aplicaciones personalizadas. Adaptar el sistema de control basado en las condiciones pre defecto. Escala de valores analógicos y cierre momentáneo sobre las entradas para la recuperación de SCADA.

- ✓ **Control de re-conexión.**

Reconectador programable de cuatro disparos con un chequeo de sincronismo opcional para que coincida con las diferentes prácticas de re-conexión.

- ✓ **Diseño resistente.**

Resistente a una amplia gama de temperatura ambiente de funcionamiento de la industria, de -40 ° a +85 ° C (-40 ° a +185 ° F).

- ✓ **Notificación automática.**

Alerta al personal los problemas de forma automática y directa mediante SEL-3010 Mensajero de eventos de apoyo.

- ✓ **Fácil instalación**.

Fácil de instalar en las ubicaciones existentes utilizando paquetes disponibles de adaptación y adaptadores de montaje, sin cortar o perforar.

- ✓ **Diseño Flexible**.

Varias maneras de instalación y las opciones de integración con un factor de forma pequeño y deslizante en las tarjetas de expansión.

5.6.1.2 Opciones:

5.6.1.2.1 Entrada - salida flexible para Control Local y Usos de Sistema.

El sistema bajo incluye tres salidas digitales y dos entradas digitales. Hay tres ranuras de tarjeta para tarjetas de entrada - salida escogidas en forma opcional, incluyendo lo siguiente:

- ✓ Cuatro entradas digitales (DI) y cuatro salidas digitales (DO)

- ✓ Cuatro entradas digitales (DI) y cuatro de interrupción actual alta y rápida de las salidas digitales (DO)
- ✓ Cuatro salidas digitales (HACEN), tres entradas digitales (DI), y una salida análoga (AO)
- ✓ Cuatro entradas análogas (AI) y cuatro salidas análogas (AO)
- ✓ Ocho entradas análogas (AI) para tratar valores DC
- ✓ Ocho entradas digitales (DI)
- ✓ Tres entradas de voltaje AC
- ✓ Diez entradas de RTD

5.6.1.3 Aplicaciones

- ✓ Personalizar la operación del botón del panel delantero y LEDS, o use funciones de muestreo de onda de transcientes.
- ✓ Personalizar mensajes LCD que usan ajustes de punto de demostración conducidos por acontecimiento.
- ✓ Crear un sistema de control integrado con una variedad de opciones de comunicaciones y entrada - salida.
- ✓ Usar la lógica de control programable y rasgos de integración con una línea de comunicaciones para el control y la protección de subestaciones remotas.

- ✓ Usar el reportaje comprensivo para entender acontecimientos, el mantenimiento de lista, descubrir tendencias desfavorables, modificar cargas, y satisfacer las exigencias de la información de sistemas de supervisión.

- ✓ Incluir entradas de RTD como la parte de integración de sistema o influir la protección.

- ✓ Analizar el funcionamiento del sistema de protección de sobre corriente que usa el Registrador de Acontecimientos empotrado Secuencial.

- ✓ Usar ACSELERATOR® QuickSet™ SEL-5030 o SEL-5010 Relay Assistant, Software para manejar sus ajustes de relevo.

- ✓ Instalar la protección donde es necesario sin recintos especiales o sistemas de ventilación. La clase 1, la División 2 de certificación, permite al SEL-751A ubicarse en las posiciones que pueden ser adyacentes a gases arriesgados, vapores, o líquidos.

5.7 Protocolo de comunicación MODBUS TCP/IP

MODBUS TCP/IP es una variante o extensión del protocolo Modbus que permite utilizarlo sobre la capa de transporte TCP/IP. De este modo, Modbus-TCP se puede utilizar en Internet, de hecho, este fue uno de los objetivos que motivó su desarrollo (la especificación del protocolo se ha remitido a la IETF=Internet Engineering Task Force). En la práctica, un dispositivo instalado en Europa podría ser direccionado desde EEUU o cualquier otra parte del mundo.

MODBUS TCP/IP se ha convertido en un estándar industrial de facto debido a su simplicidad, bajo coste, necesidades mínimas en cuanto a componentes de hardware, y sobre todo a que se trata de un protocolo abierto. En la actualidad hay cientos de dispositivos MODBUS TCP/IP disponibles en el mercado. Se emplea para intercambiar información entre dispositivos, así como monitorizarlos y gestionarlos. También se emplea para la gestión de entradas/salidas distribuidas, siendo el protocolo más popular entre los fabricantes de este tipo de componentes.

La combinación de una red física versátil y escalable como Ethernet con el estándar universal de interredes TCP/IP y una representación de datos independiente de fabricante, como MODBUS, proporciona una red abierta y accesible para el intercambio de datos de proceso.

5.7.1 TCP/IP

Estos protocolos se utilizan en conjunto y son el protocolo de transporte de Internet. Cuando la información del Modbus es enviada usando estos protocolos, los datos se pasan por TCP en el que se adjunta información adicional del Modbus y dado por IP. A continuación IP coloca los datos en un paquete o datagramas y lo transmite.

TCP debe establecer una conexión antes de transferir los datos, ya que es un protocolo orientado a la conexión. El Master (o de cliente en Modbus TCP) establece una conexión con el esclavo (o Server). El servidor espera una conexión entrante desde el cliente. Una vez que se establece una conexión, el servidor responde a las consultas del cliente hasta que el cliente cierra la conexión.

5.7.2 Protocolo Modbus TCP

Modbus/TCP simplemente encapsula una trama Modbus en un segmento TCP. TCP proporciona un servicio orientado a conexión fiable, lo que significa que toda consulta espera una respuesta. Esta técnica de consulta/respuesta encaja perfectamente con la naturaleza Maestro/Esclavo de Modbus, añadido a la ventaja del determinismo que las redes Ethernet conmutadas ofrecen a los usuarios en la industria. El empleo del protocolo abierto Modbus con TCP proporciona una solución para la gestión desde unos pocos a decenas de miles de nodos.

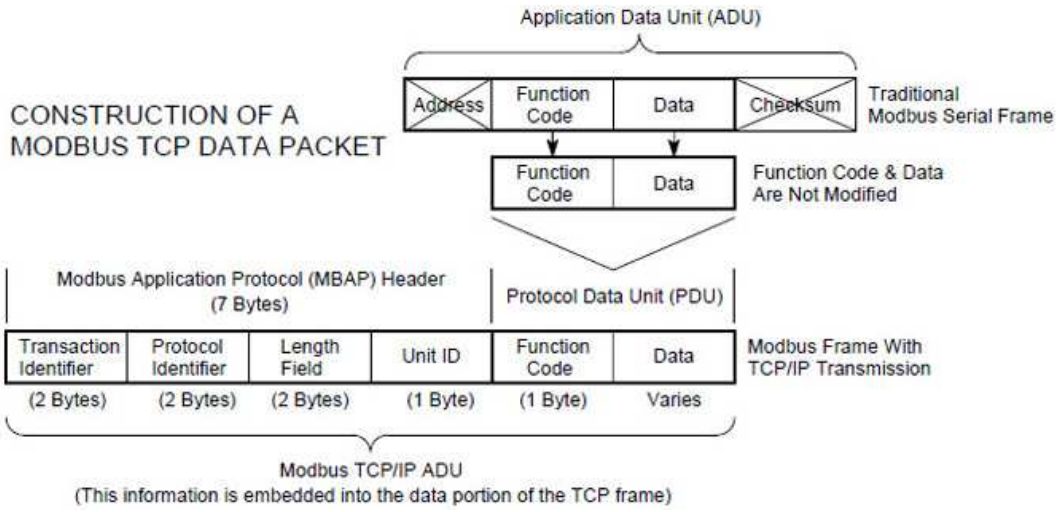


Figura V.9. ADU del protocolo Modbus TCP/IP

5.7.2.1 Características del protocolo Modbus

- ✓ Es un protocolo de transmisión diseñado por Gould Modicon (Schneider Electric).
- ✓ No tiene definida la capa 1, pero se utiliza en conexiones RS232 o RS485, utilizando UART.

- ✓ Es simple y flexible, permitiendo transferencia de datos digitales y análogos.
- ✓ Se transmite en velocidades que van desde 9600 bps a 115.000 bps. (0.115 Mbps)
- ✓ Opera en el modo maestro - esclavo, en el que existe un solo maestro.
- ✓ Los datos se comparten entre múltiples operadores de estaciones de trabajo, sistemas de control, bases de datos, que puede terminar en complejas jerarquías para compartir los datos.
- ✓ Algunas características de este formato son fijas, como por ejemplo el formato y secuencia de la trama, manejo de los errores y las condiciones de excepción.
- ✓ Cuenta con dos modos de transmisión: ASCII y RTU (hexadecimal), de los cuales el último es el más común en su uso.
- ✓ El modo RTU (modo binario o modo B) es más rápido y compacto.
- ✓ El modo ASCII (modo A), es más amplio y requiere el doble de tamaño para transmitir los mismos datos.
- ✓ Para el chequeo de errores se utiliza CRC16 con Polinomio de generación A001.
- ✓ Las tramas de Modbus pueden enviarse utilizan empaquetamiento TCP/IP.

5.7.2.2 Estructura de la trama Modbus

El formato de la trama se muestra en la siguiente tabla:

Tabla V.IX. Estructura de la Trama Modbus.

Campo de la dirección	Campo de la función	Campo de datos	Campo de chequeo del error
1 byte	**1 byte**	**variable**	**2 bytes**

- ✓ En una trama de pedido, el campo de dirección especifica la dirección del dispositivo al cual se realiza el pedido.
- ✓ La trama de respuesta inicia con la dirección del dispositivo al que se realiza el pedido.
- ✓ El valor de la dirección puede ser entre 1 y 247 (247 dispositivos).
- ✓ En aplicaciones prácticas se utilizan muy pocos esclavos.
- ✓ El segundo campo de un byte, en las tramas de pedido, especifica la función solicitada. En las de respuesta, especifica la misma función solicitada siempre y cuando, se realice lo solicitado, caso contrario se contesta con el bit más significativo de este campo en “1”.
- ✓ El tercer campo es el de datos, que varía en longitud de acuerdo a la función solicitada. En la respuesta, la longitud depende de la misma.
- ✓ El campo final es el de CRC, que realiza el cálculo en base a todos los datos de la trama.

La tabla V.X especifica las direcciones asignadas a diferentes dispositivos.

Tabla V.X. Direcciones asignadas a diferentes dispositivos.

Tipo de datos	Dirección absoluta	Dirección relativa	Código de función	Descripción
Bobina	00001 - 09999	0 - 9998	01	Lea el estado de una bobina
Bobina	00001 - 09999	0 - 9998	05	Force a una bobina
Bobina	00001 - 09999	0 - 9998	15	Force a varias bobinas
Entradas discretas	10001 - 19999	0 - 9998	02	Lea el estado de entrada
Registros de entrada	30001 - 39999	0 - 9998	04	Lea el registro de entrada
Registro intermedio	40001 - 49999	0 - 9998	03	Lea un registro intermedio
Registro intermedio	40001 - 49999	0 - 9998	06	Defina un solo registro intermedio
Registro intermedio	40001 - 49999	0 - 9998	16	Defina varios registros intermedios
----	----	----	07	Lea el estado de excepción
----	----	----	08	Prueba de diagnóstico de realimentación

5.6.3 Prestaciones de un sistema Modbus TCP/IP

Las prestaciones dependen básicamente de la red y el hardware. Si se usa MODBUS TCP/IP sobre Internet, las prestaciones serán las correspondientes a tiempos de respuesta en Internet, que no siempre serán las deseables para un sistema de control. Sin embargo pueden ser suficientes para la comunicación destinada a depuración y mantenimiento, evitando así desplazamientos al lugar de la instalación. Si disponemos de una Intranet de altas prestaciones con conmutadores Ethernet de alta velocidad, la situación es totalmente diferente.

En teoría, MODBUS TCP/IP, transporta datos hasta 250/(250+70+70) o alrededor de un 60% de eficiencia cuando se trasfieren registros en bloque, y puesto que 10 Base T proporciona unos 1.25 Mbps de datos, la velocidad de transferencia de información útil

será:

1.25M / 2 * 60% = 360000 registros por Segundo

En 100BaseT la velocidad es 10 veces mayor.

Esto suponiendo que se están empleando dispositivos que pueden dar servicio en la red Ethernet aprovechando todo el ancho de banda disponible.

5.6.4 Problemas de operación en Modbus

- ✓ A pesar de los muchos cuidados que se tenga en la implementación de una red industrial, siempre se presentarán problemas.
- ✓ Los problemas de las redes industriales se pueden dividir en problemas de hardware y problemas de software.
- ✓ Los problemas de hardware se relacionan con las conexiones y las interfaces de comunicación.
- ✓ Los problemas de software tienen que ver con el acceso a nodos no existentes, funciones no válidas, formatos no válidos, protocolo no funcionando adecuadamente, etc.
- ✓ Para la solución de problemas se requiere de equipo adecuado tanto de hardware como de software.
- ✓ En cualquier caso siempre se requiere del uso de los manuales.

- ✓ En el caso de problemas de hardware se requerirá del uso de cajas de prueba de interfaces, probadores de continuidad, voltímetros, destornilladores, alicates, cables de repuesto, etc.

- ✓ Para el caso de problemas de software, se necesitará esencialmente analizadores de protocolo, mediante analizadores dedicados (caros) o mediante el uso de simuladores de software.

- ✓ Es necesario definir en forma precisa, tanto en el maestro como en el esclavo, los parámetros de comunicación tales como el número de bits, velocidad de transmisión, bits de parada, paridad y tipo de paridad, etc.

CAPÍTULO VI

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE MONITOREO DE LOS MEDIDORES Y RELÉS DE LA S/E1 DE LA EERSA

Después de haber identificado las áreas de interés del presente proyecto, y luego de haber realizado la especificación de los instrumentos y equipos necesarios para cumplir los requerimientos mínimos de medición de los medidores de la S/E1 de la EERSA, es necesario detallar las características técnicas mínimas del sistema de supervisión y adquisición de datos (SCADA) y las interfaces Hombre-Máquina (HMI) de la S/E1.

6.1 Arquitectura del sistema SCADA

La figura VI.10 muestra el esquema general de la arquitectura del sistema a implementar en la S/E1 de la EERSA.

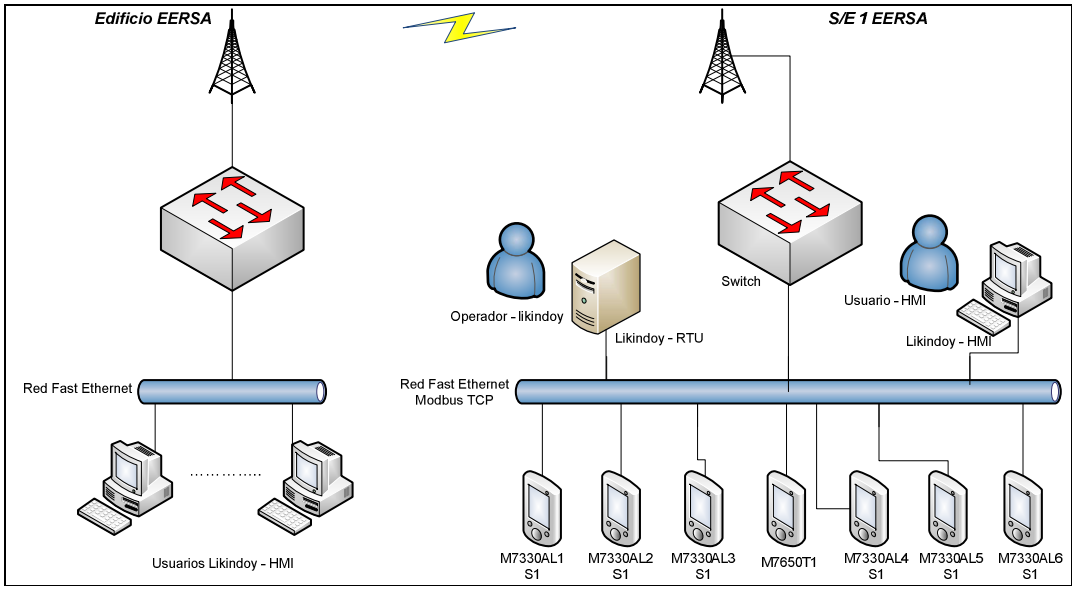


Figura VI.10. Arquitectura general del sistema SCADA

Para este proyecto se ha previsto la implementación de un sistema SCADA local, encargado del monitoreo, supervisión y almacenamiento de datos de los medidores ion 7650 e ion 7330 de los alimentadores que posee la S/E1 de la EERSA.

6.2 Elementos del sistema de monitoreo

El sistema SCADA a implementar para el monitoreo de los medidores y relés de la S/E1 de la EERSA está constituido por los siguientes elementos.

- ✓ Sistema de adquisición de datos
- ✓ Sistema de comunicaciones
- ✓ Hardware y software del SCADA
- ✓ Centro de monitoreo

6.2.1 Sistema de adquisición de datos

Una parte muy importante en los sistemas de monitorización, son los equipos de adquisición de datos, para esta aplicación se utilizara un PC con Linux Debían Lenny 5.0 que corre Likindoy-RTU, este toma los datos de los medidores ion de cada alimentador en un intervalo de 1 a 2 segundos, los almacena en un archivo de texto plano con un formato estándar (signal.dat) y luego son comprimidos mediante el programa bzip en un archivo .gz, los cuales sirven para realizar las graficas y reportes. Al mismo tiempo crea una tabla temporal por cada medidor en la base de datos de MySQL, en las cuales se almacenan los últimos datos adquiridos los cuales se muestran en la interfaz HMI.

Likindoy-RTU enlazara el nivel de campo y el nivel del centro de monitorización. La conexión con el nivel de campo se realizara mediante módulos de comunicación que soporta el protocolo industrial seleccionado para la implementación de la técnica de transmisión “paso de testigo” (bus de campo). La conexión con el nivel de monitorización se realizara mediante un switch hacia la red Ethernet, utilizando el protocolo Modbus TCP/IP.

6.2.1.1 Instalación del sistema

Likindoy SCADA libre se instalara bajo el sistema operativo Linux Debían Lenny 5.0

6.2.1.1.1 Requisitos

El sistema mínimo que likindoy necesitara para funcionar es Python, mediante este se le ira informando del resto de dependencias y carencias del sistema gracias a su instalador integrado.

No obstante los requisitos a gran escala son:

- ✓ **Un servidor MySQL accesible desde la maquina (local o remoto) y librerías para Python para acceder a MySQL.-** Almacena los datos recogidos de todas las señales.
- ✓ **Sistema R-CRAN 2 y librería rpy.-** Lenguaje de análisis estadístico R y dispone el entorno necesario para generar las graficas.
- ✓ **Librería GnuPGInterface para Python.-** Obligatorio por la librería de encriptación, permite al servidor UDP usar encriptación PGP además de la ya incluida TripleDES.
- ✓ **Librería Image para Python.-** Genera ciertas marcas especiales en la gráficas.
- ✓ **Librería SQLite para Python.-** Permite al sistema RTU almacenar localmente la información que va recibiendo del sistema así como su estado interno.
- ✓ **Librería Paramiko para Python.-** Permite descargar ficheros de datos mediante SFTP.
- ✓ **Librería XML para Python.-** Necesaria para leer los ficheros de OpenOffice Calc.

6.2.1.1.2 Instalación de MySQL

Para instalar el servidor y un cliente de líneas de comandos MySQL, se ejecuta el siguiente comando:

 # aptitude install mysql-server mysql-client mysql-common libmysqlclient15-dev

Se debe tener en cuenta que hemos instalado las librerías y los encabezados de desarrollo con el paquete “libmysqlclient15-dev”.

Durante la instalación de MySQL, se presentara la opción de establecer una contraseña:

Figura VI.11. Contraseña de root MySQL

Estableciendo la contraseña del usuario administrador “root” de MySQL es un paso recomendado en la instalación.

6.2.1.1.3 Instalación de Likindoy

Likindoy se lo puede encontrar disponible en la página oficial www.likindoy.org el cual será copiado al directorio en donde se va a ejecutar (/usr/scada-likindoy-eersa).

Existen dos maneras de instalar likindoy la instalación manual y la automática, la instalación se la hará en forma automática.

Para realizar la instalación en forma automática se deberá acudir al directorio doc/ dentro de likindoy “/usr/scada-likindoy-eersa/doc” en el que se podrá encontrar el archivo install.py (Anexo 1) que se encargara de comprobar los paquetes que faltan y de crear el usuario y base de datos correspondiente para usar likindoy.

En las figuras VI.12a, VI.12b, VI.12c, VI.12d. Se muestra el proceso de instalación de likindoy.

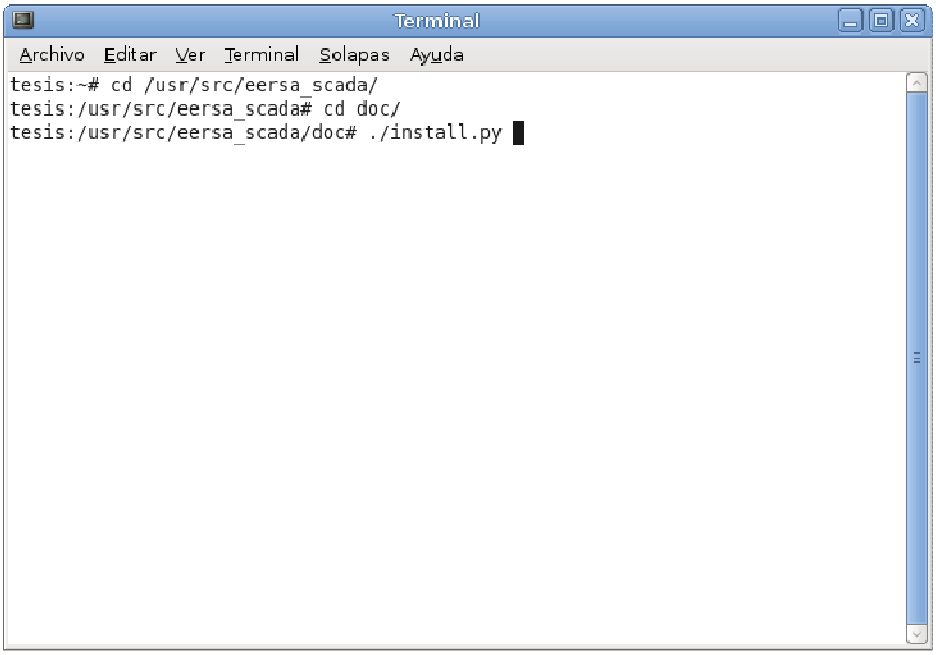


Figura VI.12a. Ejecución del programa de instalación.

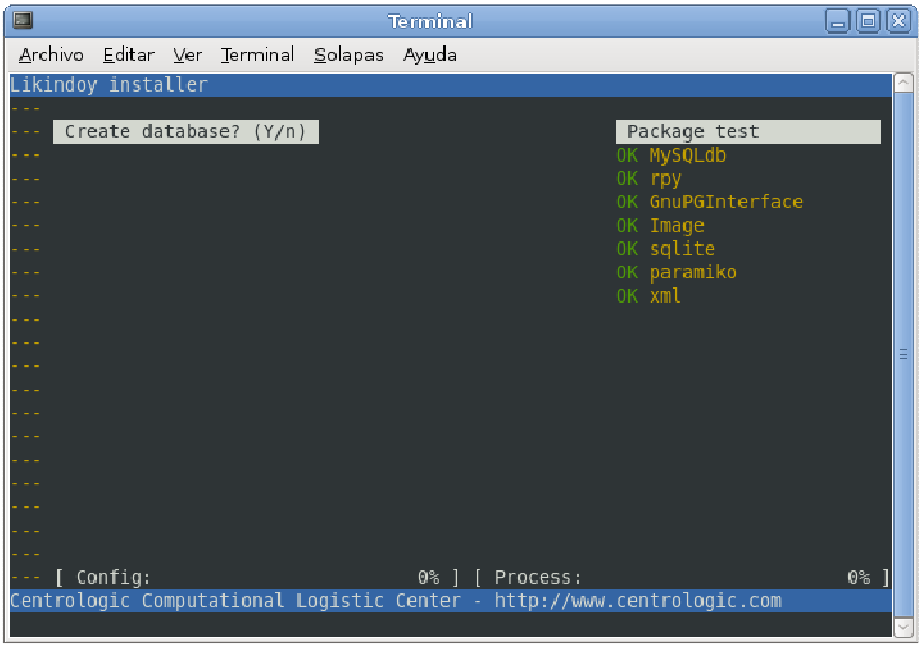


Figura VI.12b. Paquetes que requiere likindoy para la instalación.

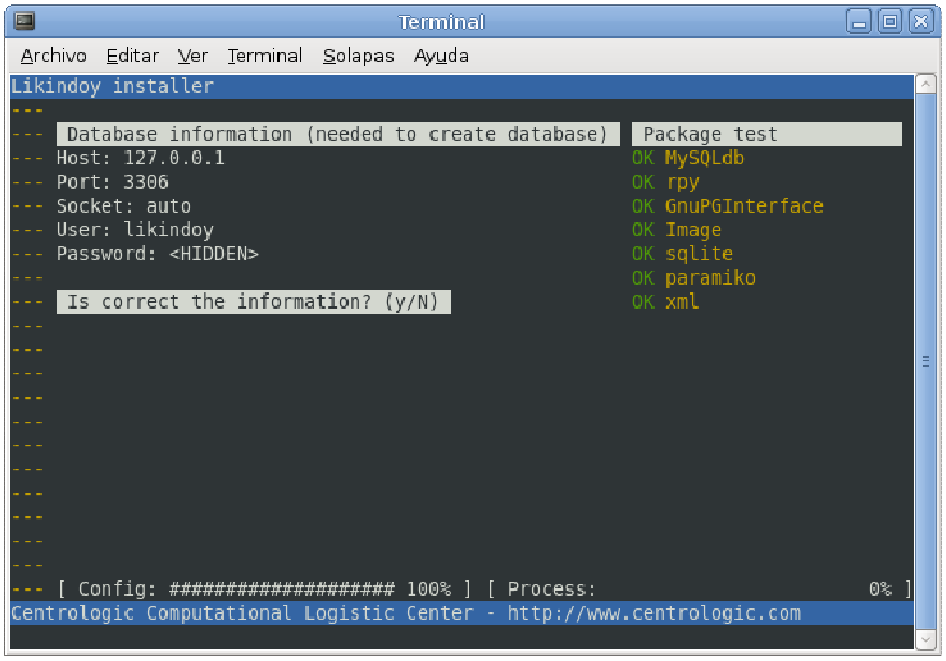


Figura VI.12c. Creación de la Base de Datos y usuario de acceso.

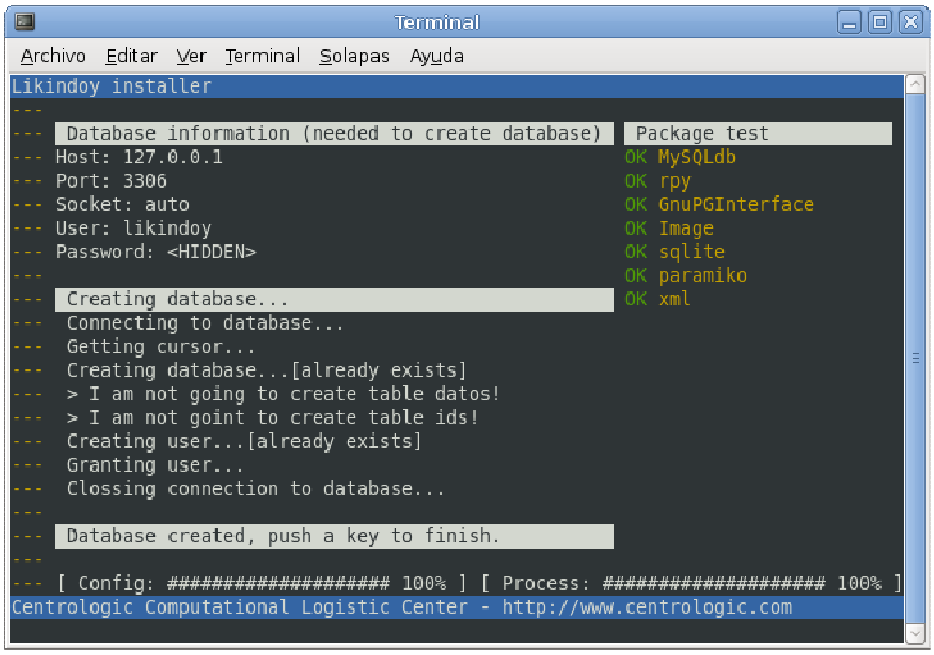


Figura VI.12d.Creación de la base de datos likindoy_test.

En la base de datos disponemos de 12 tablas que almacenan la información necesaria para que el sistema funcione adecuadamente.

- ✓ **Tabla ids:**
  - ✓ **id:** identificador de la señal cuyos datos podemos encontrar en la tabla de datos.
  - ✓ **md5:** texto md5 resultado de hacer el md5 del nombre de la señal en texto.
  - ✓ **updated:** indica la fecha epoch de la última actualización de dicho campo.
- ✓ **Tabla datos:**
  - ✓ **id:** identificador de la señal que podemos encontrar relacionado en la tabla ids.
  - ✓ **server:** fecha del servidor en el momento de la inserción.
  - ✓ **fecha:** fecha del programa en el momento de la inserción.
  - ✓ **hora:** hora del programa en el momento de la inserción.
  - ✓ **valord:** valor digital si la señal es digital.
  - ✓ **valora:** valor analógico si la señal es analógica.

- ✓ **Tabla M7650T1, M7330AL1S1, M7330AL2S1, M7330AL3S1, M7330AL4S1, M7330AL5S1, M7330AL6S1:**
  - ✓ **id:** identificador de las señales que se encuentran relacionados en la tabla ids, cada medidor cuenta con sus propios ids.
  - ✓ **valora:** valor analógico de las señales por cada medidor.
  - ✓ **Unidad:** Unidad dimensional en la cual se muestran los datos.

  NOTA: Cada tabla se crea con el nombre del telemando y contiene los datos de cada medidor.

- ✓ **Tabla nombresid:**
  - ✓ **id:** Identificador de la señal.
  - ✓ **nombreid:** Nombre de la señal.
- ✓ **Tabla rangos:** Contiene el rango de medición de cada señal.
  - ✓ **id:** Identificador de cada señal.
  - ✓ **valoramin:** Contiene el valor mínimo.
  - ✓ **valoramax:** Contiene el valor máximo.
- ✓ **Tabla seguridad:** Contiene la clave para la configuración de los Puntos de Ajuste (Setpoints).
  - ✓ **usuario:** Contiene el nombre del usuario.

- ✓ **clave:** Contiene la clave del usuario.

El usuario de likindoy únicamente recibe los datos del sistema. Esto equivale a visualizar los datos, las graficas generadas y entenderlas en la medida en que el usuario y el operador acordaron en lo referente a la simbología.

Igualmente para los usuarios de los sistemas SCADA en tiempo real el resultado viene dado directamente por las decisiones tomadas en el momento de generar la configuración del sistema y por lo tanto el operador será el encargado de indicar la simbología y significado de cada elemento del sistema.

El operador se limita únicamente a modificar los ficheros de configuración dispuestos en el directorio etc/, conoce la estructura general del sistema y por lo tanto mediante esta se encarga de definir cada uno de los parámetros.

El directorio etc/ contiene toda la configuración específica del sistema, esto es:

Telemandos.- Declaración de todos los telemandos del sistema indicando a likindoy donde debe recoger la información, claves de acceso, etc. Cada fichero de recogida se ha separado por el modo en que los datos son recogidos, de tal modo que se ha nombrado a todos como adquirir_*.py y cada cual corresponde a su propio módulo dentro de likindoy. Por ejemplo: adquirir_signals.py contiene la configuración de todos los telemandos likindoy-compatibles, el fichero adquirir_pesyr.py contiene la configuración de todos los telemandos PESYR, el adquirir_ftp.py para que podamos bajarnos los datos de ciertos telemandos directamente mediante FTP.

Cargador.py.- Contiene únicamente el orden en que deben ser cargados los ficheros, de tal modo que así podamos disponer de un conjunto de datos antes que otro aunque este

primero haya llegado después. Es muy útil para acelerar la generación de graficas específicas.

Señales.ods.- Es un fichero OpenDocument Spreadsheet (OpenOffice Calc) y contiene toda la información relativa a las señales del sistema. Indica desde su nombre interno, descripción, telemando al que pertenece, si es analógico o digital, en que conector del telemando se encuentra, como debe ser procesada en la adquisición, etc.

Otros.- Existen otros ficheros relativos a configuraciones de subprogramas de likindoy, desde gestión de ping para estadísticas de red, servidores UDP, gestión de alertas, etc.

Existe un segundo directorio de configuración supergrupo de etc/ y se encuentra en src/etcbin/. No se permite al operador tocar en este directorio, pues en él se localiza la información genérica del sistema, incluyéndose en el detalle de acceso a servidores MySQL, llaves DSA para uso de SSH, sistema de gestión de señales, etc.

Casi todas las opciones definidas por defecto en este directorio pueden ser sobrescritas mediante las configuraciones disponibles en etc/ por lo que un operador no debería tener necesidad de tocar en este directorio. El resto de opciones que no son transparentes en etc/ se refieren a configuraciones que solo el administrador debe controlar y que requieren un conocimiento más avanzado del sistema.

6.2.1.2 Configuración del sistema

El entorno de trabajo en likindoy es en modo convergente, es decir se enfoca tus esfuerzos en orden:

1. Configuración de la adquisición de datos.
2. Prueba la adquisición de datos correctamente.
3. Configuración del orden del sistema de carga.
4. Añade todas las señales que adquieres.
5. Prueba que el sistema de carga almacena la información en la base de datos.
6. Configuración de todas las gráficas.
7. Prueba que el sistema de gráficas genera los ficheros correctamente.

Asegúrese de usar mayúsculas y minúsculas adecuadamente en todo momento ya que el sistema es sensible a estas y considera cadenas distintas.

6.2.1.2.1 Configuración de la adquisición de datos

A través del medio de comunicación instalado, la estación maestra solicita y recopila la información desde los equipos ion.

Este procedimiento guarda la información recopilada en bases de datos y la pone a disposición de los operadores u otros componentes del sistema.

Para realizar la configuración de la adquisición de datos se deberá acudir al directorio /etc dentro de likindoy “/usr/scada-likindoy-eersa/etc” en el que se podrá encontrar el archivo adquirir_momentum.py que se encargara de leer los datos de medición usando la librería Modbus.

En el archivo adquirir_momentum.py (Anexo 2) se encuentra la configuración general de los medidores ion, la configuración de los registradores, divisores, unidades de cada señal, y la configuración de los telemandos.

En la configuración general de los medidores ion se especifica todas las señales o registros que se van a añadir en los registradores con su respectiva unidad y divisor para la adquisición de datos.

En el Anexo 2 se muestra la configuración general de los medidores ion.

El registrador es un elemento que se encarga de registrar o almacenar los datos en un modo concreto. En likindoy existen los FILE_SIGNALS, los NET_UDP y otros.

Los FILE_SIGNALS registran los datos en un fichero de tipo SIGNALS. Este tipo de fichero es propio de likindoy y fue diseñado pensando en la claridad de los datos y facilidad de uso. Es el tipo de fichero estándar para todos los registrados por RTU's de likindoy.

Al registrar los datos en un objeto de este tipo los datos son enviados a un servidor externo mediante paquetes UDP. Es el formato estándar usado por likindoy para exportar información en tiempo real a otros sistemas.

Es posible realizar nuevos registradores con bastante facilidad implementando el interface REGISTRADOR que se puede encontrar en la librería interfaces.

Al crear un objeto FILE_SIGNALS el primer parámetro es el formato del fichero. El formato del fichero que se va a generar depende del modelo del registrador físico con el que se va a trabajar, eligiendo un tipo u otro fichero modificamos también el grupo de

funciones aceptadas por el mismo en el momento en que se registran. Además el indicar el tipo de fichero que añade información de control extra del fichero. Puede ser compatible con los siguientes modelos de RTU's:

Advantech: Indica al registrador que se van a registrar datos de un sistema Advantech.

Momentum: Indica al registrador que se van a registrar datos de un sistema Schneider Momentum.

Netstatistics: Indica al registrador que se van a registrar datos de un sistema que está usando el modulo NETSTATISTICS de likindoy.

Web. Indica al registrador que se van a registrar datos de un sistema que está usando el modulo WEB de likindoy.

El segundo parámetro es el nombre del fichero (sin la extensión "signals.dat").

El tercer parámetro es "compress=True" para indicar al registrador que comprima el resultado (la extensión entonces será "signals.dat.gz").

Entonces estos registradores registran todos los datos que le son entregados.

En el Anexo 1 se muestra la configuración de los registradores.

El objeto REGISTRADORES requiere una lista de registradores, esto es "[objeto_registrador_1, objeto_registrador_2,..]". El objeto registradores será usado por la RTU para enviar los datos, de tal modo que este objeto se encargara a su vez de informar a cada uno de los registradores del dato entregado por la RTU.

Su diseño está separado de la RTU para evitar la sobrecarga de registrar los datos dentro de la RTU, cuya única preocupación debería ser la obtención, decisión y distribución de los datos a las distintas clases.

El objeto registradores podría ser ampliado para enviar los datos en paralelo a cada una de las clases separando cada una de estas ejecuciones en hilos diferentes.

Para configurar la adquisición de datos se utilizaran también los telemandos, la información que requieren un telemando para su configuración es:

- ✓ Primero declaramos el telemando
- ✓ Indicamos al telemando la configuración que queremos usar: dirección IP del telemando y el nombre del fichero de datos que nos queremos traer del mismo.
- ✓ Indicamos el nuevo usuario (para que no use el usuario por defecto). Este campo es opcional.
- ✓ Indicamos la nueva clave (Para que no use la clave por defecto). Este campo es opcional.
- ✓ Añade el telemando a la lista de telemandos que será procesada en el script principal de likindoy.
- ✓ Añade el objeto del telemando a una lista la cual contiene todos los telemandos del sistema.

En el Anexo 2 se muestra la configuración de todos los telemandos.

6.2.1.2.2 Prueba la adquisición de datos correctamente

Para verificar la adquisición de datos se deberá acudir al directorio “/usr/scada-likindoy-eersa/” en el que se deberá crear los archivos por cada telemando a partir del adquirir_momentum.

- ✓ adquirir_momentum_M7650T1.py (Anexo 3)
- ✓ adquirir_momentum_M7630_AL1S1.py (Anexo 4)
- ✓ adquirir_momentum_M7630_AL2S1.py (Anexo 5)
- ✓ adquirir_momentum_M7630_AL3S1.py (Anexo 6)
- ✓ adquirir_momentum_M7630_AL4S1.py (Anexo 7)
- ✓ adquirir_momentum_M7630_AL5S1.py (Anexo 8)
- ✓ adquirir_momentum_M7630_AL6S1.py (Anexo 9)

Los cuales se deben ejecutar para adquirir los datos de los autómatas.

Para la adquisición de datos se deberán ejecutar los archivos mencionados anteriormente y si la configuración de los adquirir_momentum_*.py son correctos entonces:

- ✓ Comprueba la configuración general del telemando que se encuentra en el /etc.
- ✓ Se conecta al autómata y realiza la adquisición de datos.

- ✓ Verifica la existencia de la tabla y si no la crea con el nombre del telemando, por cada adquisición en la tabla se realiza la eliminación de los datos actuales y la inserción de datos de la nueva adquisición.
- ✓ Los datos de la tabla se muestran en la interface HMI del usuario likindoy.
- ✓ Los datos adquiridos se guardan en ficheros tipo signals, ficheros que se almacenan en "/usr/scada-likindoy-eersa/data/dib"
- ✓ Si la adquisición falla los datos en la tabla se cargan con NULL y en el fichero tipo signals se carga con NET_ERROR 0

Los ficheros tipo SIGNALS son un formato propio de likindoy que cumple las características de ser abierto y libre. Este formato de ficheros registra en cada fila una señal y su estado correspondiente, de tal modo que en cada fila encontraremos una señal u otra y la captura instantánea de la misma. Existen dos tipos de filas "digitales" y "analógicas":

- ✓ **Digitales:** Contiene un total de 5 campos separados como se indica a continuación:
  - ✓ Fecha en formato texto
  - ✓ Fecha en formato epoch
  - ✓ Función
  - ✓ Posición
  - ✓ Valor actual

- ✓ **Analógicas:** Contienen un total de 9 campos separados como se indica a continuación:
  - ✓ Fecha en formato texto
  - ✓ Fecha en formato epoch
  - ✓ Función
  - ✓ Posición
  - ✓ Valor Actual
  - ✓ Valor medio
  - ✓ Valor máximo
  - ✓ Valor mínimo
  - ✓ Varianza

6.2.1.2.3 Configuración del orden del sistema de carga

Del cargador hay muy poco que decir debido a que casi todo se hace automáticamente. Cuando se ejecuta, busca los ficheros a cargar en el directorio data/reg/ y cuando termina de procesar el fichero lo borra de dicho directorio. La ejecución es selectiva en función de los ficheros que se encuentre en el directorio de modo que primero se cargan los ficheros indicados en la configuración del cargador y después el resto de ficheros por orden alfabético. Después de cada lazo de bucle se comprueba de nuevo la lista de ficheros para asegurarnos que siempre cargamos los más prioritarios.

El orden de ejecución de los ficheros se define dando el nombre de los telemandos en el orden en que deseamos sean cargados sus ficheros.

6.2.1.2.4 Añade todas las señales que adquieres.

El fichero de señales es el más importante de todo likindoy debido a que interconecta al "mundo real" con el "mundo virtual", de tal modo que prácticamente refleja toda la base de conocimiento del sistema.

La tabla de datos de señales.ods contiene los siguientes campos:

- ✓ **Descripción:** De la señal, no tiene ningún uso especifico en likindoy salvo aclarar la fila en la hoja de cálculo.
- ✓ **A/D:** Indica si la señal es analógica (a) o digital (d).
- ✓ **Frecuencia:** Es la frecuencia de recogida de la señal, de tal modo que el sistema interpretara que la compresión de los datos quede venir hasta con una frecuencia máxima de la indicada aquí, si pasan más segundos, minutos, horas o días de los indicados en este campo y el fichero de datos no contiene ninguna auto frecuencia a la frecuencia entonces el cargador automáticamente indicara que faltan datos en el rango implicado. Esto no afecta a la carga aunque si al generar las graficas.
- ✓ **BD:** Indica si esta señal deberá almacenarse en la Base de Datos. Es especialmente útil para señales que deseamos tener en las señales.ods para ser procesadas por otro subsistema, pero no la queremos almacenar en la base de datos.

- ✓ **RTU:** Es el nombre del telemando al que pertenece la señal.

- ✓ **ID:** Es un identificador único para cada señal. Si usted repite el identificador en 2 señales, todos los sistemas le avisaran de ello: cargador, generador de graficas, sistema de envío de emails, gestor de RTU, servidor UDP y cualquier modulo que use el fichero de señales.ods mediante la librería de señales.

- ✓ **Dirección, zona y variante:** El primero contiene la dirección física de la señal, mientras que el segundo contiene la zona de la que se obtiene los datos dentro de esa dirección. Todas las señales reales tienen una dirección física de acceso (solo las señales virtuales generadas por realimentación no tiene una dirección física, lo cual es lógico), dependiendo del tipo de señal la dirección tendrá un formato u otro, también cada tipo de dirección tiene su subconjunto de zonas. De tal modo que:

    - ✓ **Schneider – Momentum:** Usa un formato numérico compatible con la dirección propia asignada por dichos sistemas. Un ejemplo de dirección es 400413 para una señal que contiene un valor real o 100043 para una señal digital.

    - ✓ **Advantech – ADAM 5000 TCP:** Usa un sistema de direccionamiento basado en slot y posición dentro del slot. Un ejemplo de dirección es 2.0 que se refiere al slot 2 primer conector (conector 0). Este tipo de direcciones permite 2 tipos de zonas “ra” (lectura analógica *) y “rd” (lectura digital *).

- ✓ **Netstatistic:** Para el sistema de análisis de red la dirección puede ser el dispositivo sobre el cual se desea analizar la información o la dirección IP sobre la que se quiere operar, cada cual necesitara además indicar la zona desde la que se va a extraer los datos. Un ejemplo seria 192.168.0.23, también eth0 o incluso www.eersa_scada.com. Este tipo de direcciones permite varios tipos de zonas.

En el Anexo 10 se muestra el archivo señales.ods de todas las señales que se requiere para el monitoreo de los medidores ion.

6.2.1.2.5 Prueba que el sistema de carga almacena la información en la base de datos

Para verificar el sistema de carga de la adquisición de datos se deberá acudir al directorio "/usr/scada-likindoy-eersa/" en el que se podrá encontrar el archivo cargar.py que se encargara de cargar los datos adquiridos a través de los archivos adquirir_momentum_*.py en la base de datos. Entonces se deberá hacer una consulta de datos en la base de datos likindoy_test para verificar que los datos se han almacenado correctamente.

Antes de ejecutar cualquier comando vamos a renovar la lista de tareas por hacer para así asegurarnos que likindoy acepta nuestras peticiones.

./renovar.py (Anexo 11)

Si en este paso se produce algún error significa que existe algún problema de configuración.

Cargamos los datos a la Base de Datos.

./cargar.py (Anexo 12)

6.2.1.2.6 Configuración de todas las graficas

Para realizar la configuración de todas las graficas se deberá acudir al directorio /etc dentro de likindoy "/usr/scada-likindoy-eersa/etc" en el que se podrá encontrar el archivo graficas.py en la que se configuran todas las graficas que se desean obtener.

Declaración de la clase que nos proporciona los rangos y las fechas para las graficas.

```
rangos=datos_actuales ():
```

Llama a la función maxymin en la cual entrega en la variable valores una lista con el valor máximo y mínimo de las mediciones de la señal a cual corresponde el id.

```
valores=rangos.maxymin (id)
```

La función fecha actual que entrega una cadena con la fecha inicial de día y la fecha actual del sistema.

```
fechas=rangos.fecha_actual ()
```

Se genera la grafica entregando el nombre "Ia ION7650" y el objeto que contiene toda la información de las señales "s".

```
g=GRAFICA ("Ia ION7650",s)
```

Cargamos la configuración por defecto a la grafica.

```
g.config(config)
```

Numero de leyendas por fila

```
g.leyendaCulumna(5)
```

Declaramos el tamaño del eje Y como va a crecer la rejilla, indicamos que ira de -10 a 180, poniendo líneas horizontales cada 10 unidades y poniendo marca cada 5 unidades,

```
g.sizey(valores[0],valores[1],10,10)
```

Declaramos el tamaño del eje X, indicamos que irá desde el 08/11/2010 a las 8:00 am hasta 09/11/2010 a las 7:50 am y que el corte del eje X se hará por horas.

```
g.sizex(fechas[0],fechas[1],"%d/%m/%y")
```

Añade el texto del eje izquierdo como “Corriente I”

```
g.addaxis(0,"izq","Corriente I")
```

Añade el texto del eje inferior como “Tiempo (Horas)”

```
g.addaxis(0,"inf","Tiempo (Horas)")
```

Información sobre qué datos realizar el test de información disponible en la grafica. La correspondencia de cada campo es:

- ✓ Nombre de la señal sobre la que hacer el test
- ✓ minX desde que posición en X realizar el test
- ✓ maxX hasta que posición en X realizar el test
- ✓ minY desde que posición en Y realizar el test

- ✓ maxY hasta que posición en Y realizar el test
- ✓ línea. indica si el nombre dado en el primer parámetro es una línea (Por defecto False)

El test debe encontrar al menos un dato en la caja definida por las cuatro esquinas: minX, maxX, minY, maxY.

```
g.test(“M7650T1_Ia”,fechas[0],fechas[1],-100000,100000)
```

“addline” añade líneas a la grafica. El orden en que se añaden las líneas afectara al orden en que son pintadas de tal modo que la última añadida quedara por encima de las demás. La correspondencia de cada campo es:

- ✓ Nombre de la línea
- ✓ Ecuación que se mostrara
- ✓ Indica si se usara una representación analógica (a) o digital (d)
- ✓ Representación a usar para la línea o equivalente
- ✓ Color principal
- ✓ Grosor de la línea
- ✓ Tipo de línea a usar o equivalente

```
g.addline("ion7650 T1 Corriente Fase A",”M7650T1_Ia”,"a",escalera,color_caudal,1,solida)
```

Indica los nombres de las líneas que van a aparecer en la leyendas, las leyendas aparecerán en el orden en que las líneas hayan sido declaradas.

g.leyenda(“Corriente Fase A ION 7650”)

Declara la información de salida de la grafica, en este caso se indica el nombre del fichero (donde %iayer se traduce como la fecha inversa de ayer año-mes-día), el titulo de la grafica (donde %ayer se traduce como la fecha en orden normal día-mes-año), y los dos siguientes campos corresponden a la resolución horizontal y vertical.

g.outpng(“ion7650_T1 Corriente Fase A - %iayer.png”,”ion7650_T1_Corriente Fase A - %ayer”,res_otra,700)

Finalmente añadimos la grafica al listado de graficas que se procesaran en el script principal:

graficas.append(g)

Crea la carpeta en el directorio /data/dib y almacena en la carpeta creada la grafica.

paths.append(nombre de la carpeta en donde se guardaran las graficas)

En el Anexo 13 se muestra la configuración de todas las graficas del sistema en el archivo /etc/graficas.py.

6.2.1.2.7 Prueba que el sistema de graficas genera los ficheros correctamente

Para verificar que el sistema de graficas genera los ficheros correctamente se deberá acudir al directorio “/usr/scada-likindoy-eersa/” en el que se podrá encontrar el archivo graficar.py que se encargara de crear las graficas.

Al ejecutar ./graficar.py se deben crear las graficas en los siguientes directorios:

- ✓ ./data/dib/directorio creado para las graficas.
- ✓ ./data/dib.bak/ : Copia de seguridad
- ✓ ./src/tmp/dib/hist/ : Para históricos (se procesan en una web u otro programa)

Para volver a pintar es necesario ejecutar el comando antes de crear las graficas:

> ./renovar.py

6.2.2 Sistema de comunicaciones

Las señales de campo provenientes de los equipos de medición serán recogidas por el likindoy-RTU, mediante una red de campo que soporta protocolo industrial Modbus TCP/IP. Likindoy-RTU realizara el procesamiento de la información para posteriormente transmitirlo a las estaciones de monitorización mediante la utilización de una red de área local (LAN) de alto rendimiento que soporta protocolos utilizados por los entes de regulación.

El sistema de comunicación deberá permitir comunicar el centro de monitorización con cualquiera de los medidores de cada una de los alimentadores.

6.2.2.1 Protocolo de comunicación

Además de los medios físicos para la comunicación entre componentes del sistema SCADA, son necesarios también los protocolos de comunicación, que en su forma más sencilla se podrían definir como el “lenguaje informático” que hablan los componentes al momento de interactuar. Dentro del sistema de distribución eléctrica se pueden usar

distintos protocolos a diferentes niveles pudiendo ser estos de dominio público (abiertos) o pertenecientes a alguna compañía (propietarios).

6.2.2.2 Modbus

Es un protocolo abierto del modelo OSI, utilizado en comunicaciones vía módem – radio, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado para comunicar PLC cubriendo distancias considerables, especialmente en las industrias donde se ha convertido en un estándar de facto. Actualmente existen versiones del protocolo Modbus para puerto serie y Ethernet (Modbus/TCP)

Cada paquete Modbus consta de cuatro campos:

- ✓ Campo de dirección del esclavo
- ✓ Campo Función
- ✓ Campo de datos
- ✓ Campo de comprobación de errores (checksum)

6.2.2.2.1 Campo de dirección del Esclavo

El campo de dirección del esclavo de un paquete Modbus es de un byte de longitud y el único identificador del dispositivo esclavo que participan en la transacción. El rango de direcciones válidas está entre 1 y 247. Un dispositivo esclavo ejecuta el comando especificado en el paquete cuando recibe un paquete de petición con la dirección del campo esclavo partiendo de su misma dirección. Un paquete de respuesta generada por el esclavo tiene el mismo valor en el campo dirección del esclavo.

6.2.2.2.2 Campo Función

El campo función de un paquete de petición de Modbus tiene un byte de longitud y dice a donde está dirigido el esclavo con una función a realizar. Del mismo modo, el campo función de un paquete de respuesta del máster indica que función está dirigida al esclavo. En la Tabla VI.XI se muestra las Funciones Modbus soportadas por el medidor como esclavo.

Tabla VI.XI. Funciones que soporta el medidor como esclavo.

FUNCIÓN	SIGNIFICADO	ACCIÓN
03	Lectura de registros	Obtiene el valor actual en uno o más registros que contiene el medidor.
16	Registros múltiples	Lugares específicos en una serie consecutiva de registros que contiene el medidor. Los registros explorados que pueden escribir en el medidor se muestran en el mapa de registros.

Fuente: Protocol Document Schneider Electric.

Función 3 Lectura de Registros.- Para leer los valores de parámetro el dispositivo maestro debe enviar al esclavo un paquete de petición de registros.

Para el paquete de petición de lectura de registros se especifica el inicio del registro y un número de registro para leer. El inicio del registro es numerado desde cero (40001 = cero, 40002 = uno, etc.).

El medidor responde con un paquete que contiene los valores del registro en el rango definido en la petición.

6.2.2.2.3 Campo de Datos

El campo de datos de una solicitud de Modbus es de longitud variable, y depende de la función. Este campo contiene la información requerida por el dispositivo esclavo para ejecutar el comando especificado en un paquete de solicitud o de datos que se pasa de nuevo por el esclavo en un paquete de respuesta.

Los datos de este campo se encuentra en registros de 16 bits, se transmiten en el orden de primer byte de orden alto, segundo byte de orden bajo.
Un registro de 16 bits contiene el valor 12AB hex. Este registro se transmite:

Byte de orden alto = 12 hex

Byte de orden bajo= AB hex

Este registro se transmite en el orden de 12 AB.

6.2.2.2.4 Campo de error Check (Checksum)

El campo de control permite que el dispositivo receptor determine si un paquete está dañado con errores de transmisión. En el modo Modbus RTU, se utiliza 16 bits de redundancia cíclica Check (CRC-16)

El dispositivo emisor calcula un valor de 16 bits, basado en cada byte en el paquete, utilizando el algoritmo CRC-16. El valor calculado se inserta en el campo de error check.

El dispositivo receptor realiza el cálculo, sin el campo de verificación de error, en el paquete completo que recibe. El valor resultante se compara con el campo de verificación de error.

En la tabla se muestra la estructura del paquete de lectura de registros.

Tabla VI.XII. Estructura del paquete de lectura de registros.

Paquete de petición de lectura de registros (Master a Esclavo)	**Paquete de respuesta de lectura de registros (Esclavo a Master)**
Unit ID / Dirección Esclavo (1 byte)	Unit ID / Dirección Esclavo (1 byte)
03 (Código de función) (1 byte)	03 (Código de función) (1 byte)
Registro de inicio (sr) (2 bytes)	Numero de bytes (2 x nr) (1 byte)
# de registros para leer (nr) (2 bytes)	Primer registro en el rango (2 bytes)
CRC Checksum	Segundo registro en el rango (2 bytes)
	…….
	CRC Checksum (2 bytes)

Fuente: Protocol Document Schneider Electric.

Petición:

Este comando es petición de contenido de salidas analógicas de registros del #40108 al 40110 del dispositivo esclavo con dirección 17.

Tabla VI.XIII. Paquete de petición de registros.

Slave	**Function**	**Start Register ()**		**# of Registers ()**		**CRC Checksum**	
11	03	00	6B	00	03	76	87

Fuente: Original.

11: Dirección esclavo (17 = 11 hex)

03: Código de función (Lectura analógica de salida de registros)

00 6B: Dirección de datos del primer registro solicitado. (40108 – 40001 = 107 = 6B hex)

00 03: Número total de registros solicitados. (3 registros 40108 a 40110)

76 87: CRC (Cyclic redundacy check) para la comprobación de errores.

Respuesta:

Tabla VI.XIV. Paquete de respuesta de registros.

Slave	Function	Byte Count	Register 1		Register 2		Register 3		CRC Checksum	
11	03	06	AE	41	56	52	43	40	49	AD

Fuente: Original.

11: Dirección esclavo (17 = 11 hex)

03: Código de función (Lectura analógica de salida de registros)

06: Número de bytes de datos a seguir (3 registros x 2 bytes cada uno = 6 bytes)

AE 41: El contenido del registro 40108

56 52: El contenido del registro 40109

43 40: El contenido del registro 40110

49 AD: CRC (cyclic redundancy check)

En la tabla VI.XV se muestran los ids de los registros que se van a leer por cada medidor.

Tabla VI.XV. IDs de los registros a leer por cada medidor.

SEÑALES / ID	M7650T1	M7330AL1S1	M7330AL2S1	M7330AL3S1	M7330AL4S1	M7330AL5S1	M7330AL6S1
VIna	40167	40011	40011	40011	40011	40011	40011
VInb	40169	40012	40012	40012	40012	40012	40012
VInc	40171	40013	40013	40013	40013	40013	40013
VInavg	40173	40014	40014	40014	40014	40014	40014
VIIab	40177	40015	40015	40015	40015	40015	40015
VIIbc	40179	40016	40016	40016	40016	40016	40016
VIIca	40181	40017	40017	40017	40017	40017	40017
VIIavg	40183	40018	40018	40018	40018	40018	40018
Ia	40150	40019	40019	40019	40019	40019	40019
Ib	40151	40020	40020	40020	40020	40020	40020
Ic	40152	40021	40021	40021	40021	40021	40021
Iavg	40155	40022	40022	40022	40022	40022	40022
Freq	40159	40025	40025	40025	40025	40025	40025
KWa	40199	40028	40028	40028	40028	40028	40028
KWb	40201	40030	40030	40030	40030	40030	40030
KWc	40203	40032	40032	40032	40032	40032	40032
Kwtot	40205	40034	40034	40034	40034	40034	40034
KVARtot	40215	40042	40042	40042	40042	40042	40042
KVAtot	40225	40050	40050	40050	40050	40050	40050
PFsigntot	40265	40058	40058	40058	40058	40058	40058

Fuente: Original.

6.2.3 Hardware y software del SCADA

Después de realizar un análisis preliminar de la situación de la S/E1 de la EERSA, se pudo observar que la mayoría ya disponen de un sistema SCADA local encargado del monitoreo y supervisión de los medidores ion de cada uno de los alimentadores; por lo tanto es recomendable que el nuevo sistema se integre a la arquitectura de monitorización y supervisión existente en la S/E1 de la EERSA

De manera general, es necesario disponer de por lo menos un servidor para la gestión del nuevo sistema y una estación de ingeniería para la implementación del HMI encargada de la supervisión.

Las características generales de los equipos a utilizar para la gestión del nuevo sistema son los siguientes:

- ✓ Computador
    - ✓ Procesador Pentium IV o superior
    - ✓ Sistema Operativo: Linux Debian Lenny 5.0
    - ✓ Disco duro 40 GB hard drive SATA de 3,5”, 7200 RPM o superior
    - ✓ Memoria 3GB o superior
    - ✓ DVD R/W: CD-RW/DVD-ROM
    - ✓ Puertos Ethernet.
    - ✓ Teclado USB en español latinoamericana

- ✓ Mouse Óptico de dos botones con Scroll
- ✓ Puertos: 4 USB

✓ Switch

- ✓ D-Link Des – 1016D
- ✓ 16 puertos 10/100 Fast Ethernet
- ✓ Alimentación 110V

6.2.3.1 Interface Hombre Maquina (HMI)

El objetivo general del desarrollo de una interface hombre-máquina para la supervisión y monitoreo es el permitir al operador una visión amplia de los procesos de medición de cada uno de los alimentadores de la S/E1. Facilitando un monitoreo en tiempo real de las variables de medición recolectadas desde los dispositivos de medición.

Estos diagramas de representación pueden mostrarse en distintas apariencias y pueden consistir en gráficos de líneas y símbolos esquemáticos para representar los elementos del proceso, o pueden consistir en fotografías digitales de los equipos sobre los cuales se animan los procesos.

6.2.3.1.1 Características generales del HMI

La interface hombre-máquina deberá presentar en tiempo real los datos de cada medidor de la S/E1 de la EERSA, es importante que los datos presentados sean los correctos, de modo que asegure un monitoreo seguro de los datos.

El modulo de tendencias debe permitir manejar tendencias tanto en tiempo real como históricas. Los reportes serán similares a los que se manejan en la actualidad de manera manual, se dejara sistematizado para evitar levantar un reporte de forma manual.

Las pantallas desarrolladas deberán tener concatenación si es necesario y ser consistentes entre sí de modo que existan las mismas directivas en todas las pantallas.

El sistema deberá ser amigable con el usuario, que posibilite la comprensión por parte del personal de planta con conocimientos medios de programación.

En el sistema se deberán implementar las siguientes pantallas:

- ✓ Visión general de la planta, visión de los diferentes medidores de cada alimentador, el diagrama permitirá que el operador ubique rápidamente al medidor.
- ✓ Visión general de cada área de medición de interés del presente proyecto, como cada uno de los alimentadores que posee la S/E1 y el totalizador.
- ✓ Pantallas de generación de reportes.
- ✓ Pantallas de generación de graficas de los diferentes parámetros medidos por el sistema, como graficas de corriente, voltaje, potencia.
- ✓ Despliegue de pantalla de acceso con userid y password para permitir al usuario acceder a setear o modificar los valores máximos y mínimos de cada uno de los parámetros de medición.

- ✓ Despliegue del punto de ajuste, para configurar el rango de valores máximos y mínimos normales que deben estar cada una de las mediciones que se realizan en el sistema, como la corriente, voltaje, potencia, etc.

Para la implementación de las pantallas del sistema HMI, en este caso se selecciono hacer mediante programación php Macromedia Dreamweaver 8.0.

A continuación se realiza una descripción más detallada de las funciones de cada una de las pantallas de usuario desarrolladas.

6.2.3.1.2 Pantalla principal

La figura VI.13 presenta la pantalla principal del sistema desarrollada para el monitoreo de los relés y medidores para la S/E1 de la EERSA mediante software libre SCADA.

La pantalla principal mostrara una visión general del sistema SCADA, ubicando cada uno de los alimentadores a realizar el monitoreo.

La pantalla presenta una fácil navegación a través de la interface y que sirven de ayuda al operador.

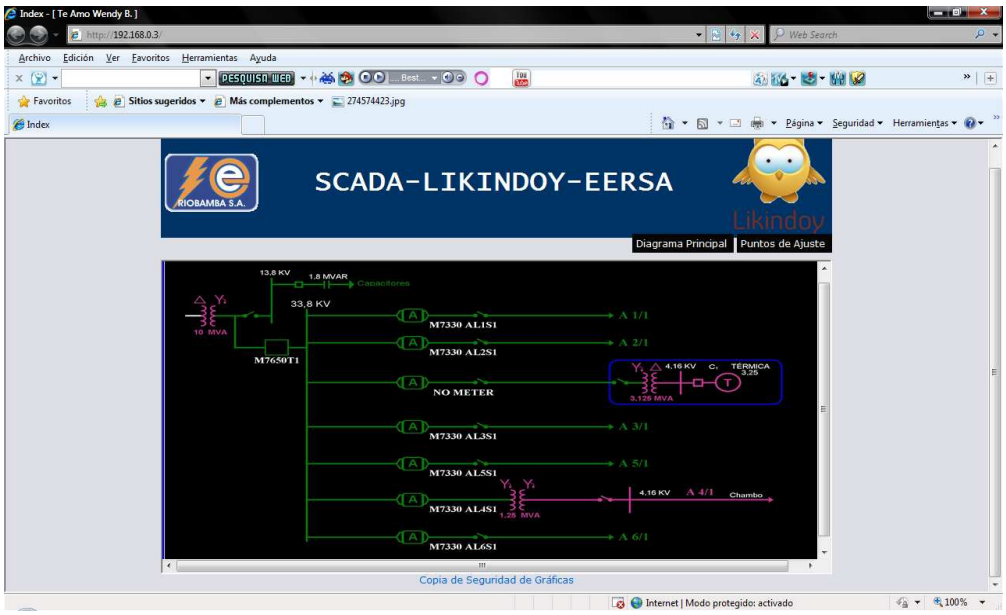


Figura VI.13. Pantalla principal del Likindoy-HMI

Al pulsar sobre cada uno de los alimentadores (Figura VI.14) se despliega una pantalla en la cual se puede visualizar los datos del medidor, además nos presenta un menú para generar las graficas, reportes de voltajes, corriente y potencias de un periodo de tiempo determinado.

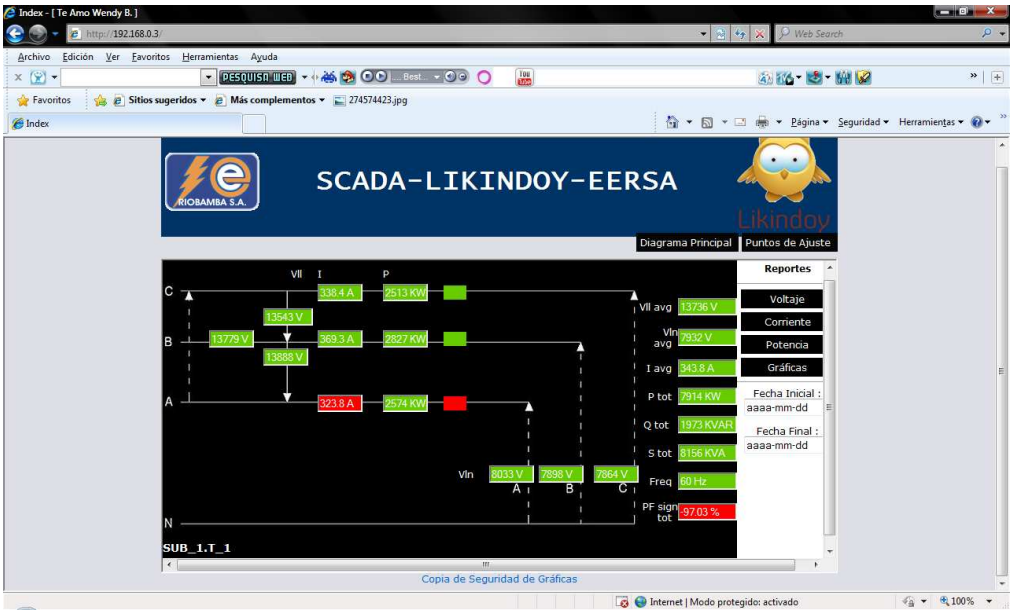


Figura VI.14. Pantalla de datos.

Los botones de reportes ya sea de voltaje, corriente o potencia, son los encargados de la generación de los reportes en formato de archivo tipo .xls (Microsoft Office Excel 2007).

Como requisito mínimo en la presentación de reportes por parte de los agentes generadores, se debe presentar lo siguiente:

- ✓ Fecha inicial y
- ✓ Fecha final.

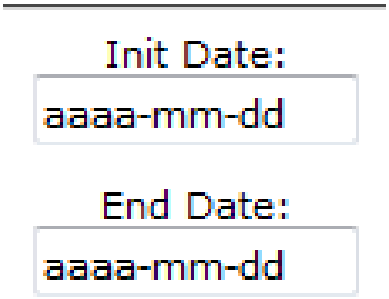


Figura VI.15. Ingreso de fechas para los reportes.

Si no se ingresa ninguno de estos parámetros el sistema no podrá sacar ningún reporte y mostrara el siguiente error:

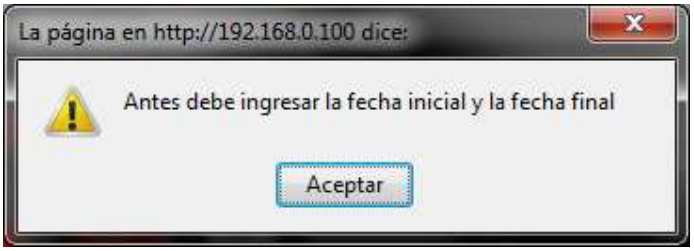


Figura VI.16. Ventana de error del reporte.

Con los parámetros ingresados correctamente al momento que el sistema genera los reportes presentara opciones de abrir o guardar el archivo tal como se muestra en la figura VI.17.

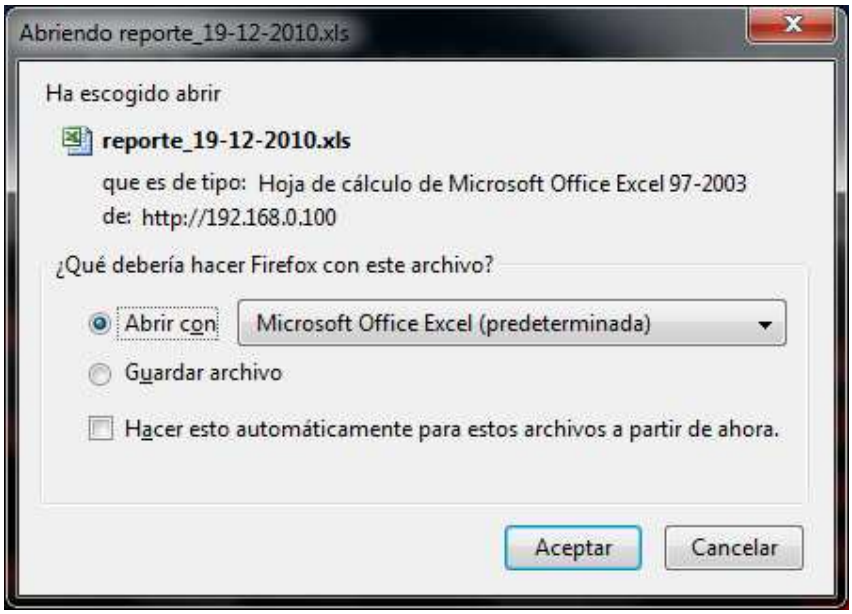


Figura VI.17. Opciones de abrir o guardar el archivo.

El usuario podría abrir directamente o bien guardar para luego abrir el reporte, al momento que usuario está abriendo el archivo se le presentara la advertencia de la figura VI.18. la cual el usuario deberá dar clic en si para poder abrirlo.

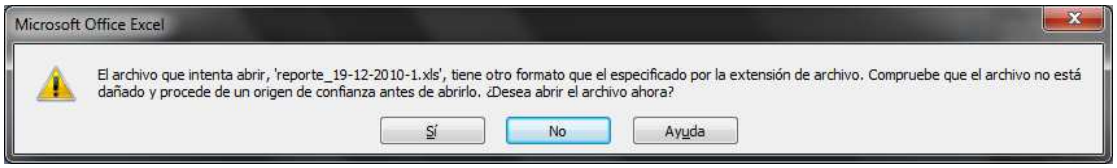
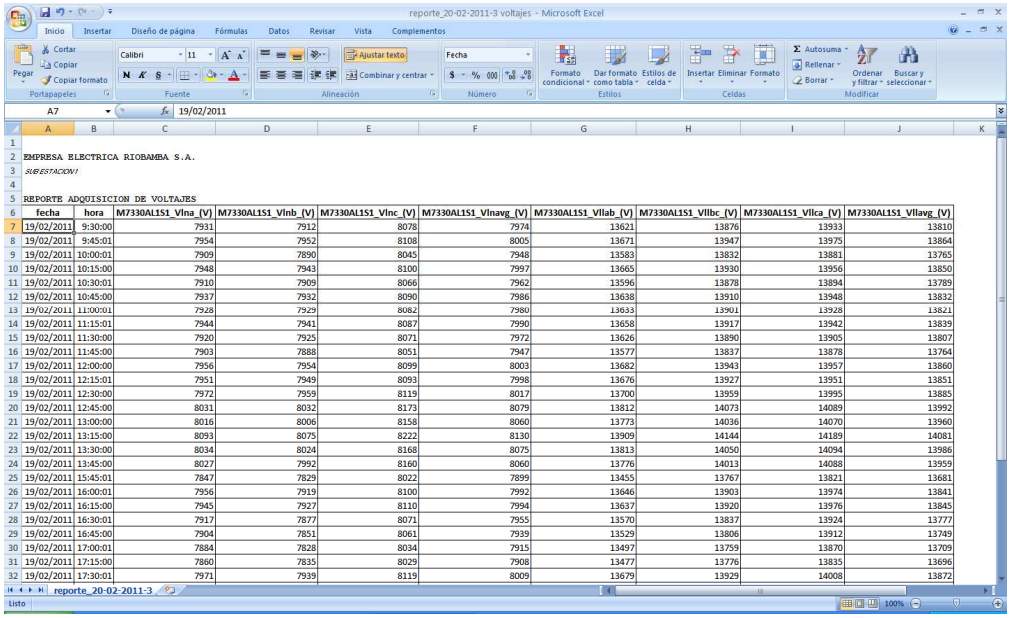


Figura VI.18. Advertencia al momento de abrir el archivo.

El reporte que presentara el sistema se muestra en la figura VI.19.



reporte_20-02-2011-3 voltajes - Microsoft Excel

EMPRESA ELECTRICA RIOBAMBA S.A.

SUBESTACION I

REPORTE ADQUISICION DE VOLTAJES

fecha	hora	M7330AL1S1_Vlna_(V)	M7330AL1S1_Vlnb_(V)	M7330AL1S1_Vlnc_(V)	M7330AL1S1_Vlnavg_(V)	M7330AL1S1_Vllab_(V)	M7330AL1S1_Vllbc_(V)	M7330AL1S1_Vllca_(V)	M7330AL1S1_Vllavg_(V)
19/02/2011	9:30:00	7931	7912	8078	7974	13621	13876	13933	13810
19/02/2011	9:45:01	7954	7952	8108	8005	13671	13947	13975	13864
19/02/2011	10:00:01	7909	7890	8045	7948	13583	13832	13881	13765
19/02/2011	10:15:00	7948	7943	8100	7997	13665	13930	13956	13850
19/02/2011	10:30:01	7910	7909	8066	7962	13596	13878	13894	13789
19/02/2011	10:45:00	7937	7932	8090	7986	13638	13910	13948	13832
19/02/2011	11:00:01	7928	7929	8082	7980	13633	13901	13928	13821
19/02/2011	11:15:01	7944	7941	8087	7990	13658	13917	13942	13839
19/02/2011	11:30:00	7920	7925	8071	7972	13626	13890	13905	13807
19/02/2011	11:45:00	7903	7888	8051	7947	13577	13837	13878	13764
19/02/2011	12:00:00	7956	7954	8099	8003	13682	13943	13957	13860
19/02/2011	12:15:01	7951	7949	8093	7998	13676	13927	13951	13851
19/02/2011	12:30:00	7972	7959	8119	8017	13700	13959	13995	13885
19/02/2011	12:45:00	8031	8032	8173	8079	13812	14073	14089	13992
19/02/2011	13:00:00	8016	8006	8158	8060	13773	14036	14070	13960
19/02/2011	13:15:00	8093	8075	8222	8130	13909	14144	14189	14081
19/02/2011	13:30:00	8034	8024	8168	8075	13813	14050	14094	13986
19/02/2011	13:45:00	8027	7992	8160	8060	13776	14013	14088	13959
19/02/2011	15:45:01	7847	7829	8022	7899	13455	13767	13821	13681
19/02/2011	16:00:01	7956	7919	8100	7992	13646	13903	13974	13841
19/02/2011	16:15:00	7945	7927	8110	7994	13637	13920	13976	13845
19/02/2011	16:30:01	7917	7877	8071	7955	13570	13837	13924	13777
19/02/2011	16:45:00	7904	7851	8061	7939	13529	13806	13912	13749
19/02/2011	17:00:01	7884	7828	8034	7915	13497	13759	13870	13709
19/02/2011	17:15:00	7860	7835	8029	7908	13477	13776	13835	13696
19/02/2011	17:30:01	7971	7939	8119	8009	13679	13929	14008	13872

Figura VI.19. Reporte en archivo tipo .xls (Microsoft Office Excel 2007).

6.2.3.1.3 Pantalla de puntos de ajuste.

El sistema permite setear los parámetros máximos y mínimos de cada señal a mostrar en la interface HMI, este seteo de datos nos permitirá realizar un control para la generación de las alarmas. Para acceder a configurar los puntos de ajuste se debe pulsar sobre el botón Puntos de Ajuste (Setpoints) en la pantalla de datos del sistema en el cual se despliega la pantalla de autenticación.

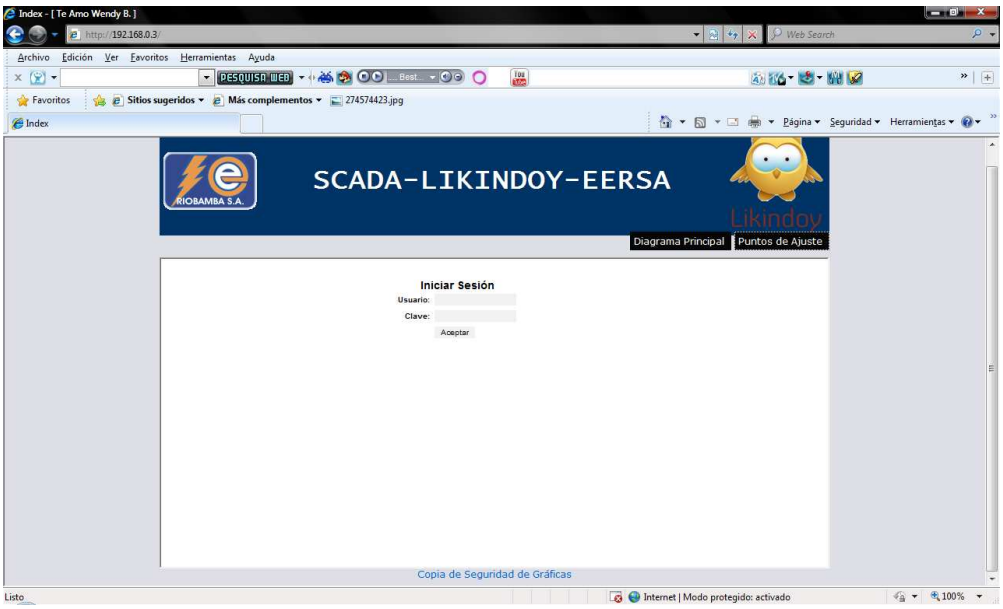


Figura VI.20. Pantalla de acceso a los puntos de ajuste.

Si el usuario administrador requiere cambiar la clave lo puede hacer en la pantalla principal de acceso a la configuración de los puntos de ajuste.

Si la autenticación fue correcta el sistema despliega la pantalla principal (figura VI.13), la cual permite seleccionar al medidor que se va a configurar.

En la figura VI.21. Se muestra la configuración de los puntos de ajuste de los datos que son mostrados en la interface y de las cuales se generan las alarmas.

Figura VI.21. Configuración de los puntos de ajuste.

Para que los cambios surjan efecto se debe guardar la configuración.

6.2.3.1.4 Pantalla de alarmas

Las alarmas se producen cuando el valor de las señales sobrepasan el umbral configurado en los puntos de ajuste, en la figura VI.22 se muestra la interface de las alarmas, en donde el color verde significa que está dentro del rango normal, y el color rojo representa una alarma la cual me indica que esta sobrepasando el umbral máximo de funcionamiento normal del sistema.

Figura VI.22. Pantalla de alarmas

6.2.3.1.5 Pantalla de gráficas

En la pantalla de graficas (Figura VI.23) se presenta un listado de todas la graficas generadas por el sistema, estas graficas se generan cada hora y contienen los datos de todo el día. Las graficas son generadas en formato jpg.

Figura VI.23. Pantalla de gráficas.

En la figura VI.24 se muestra el resultado de las graficas generadas por el sistema en formato jpg.

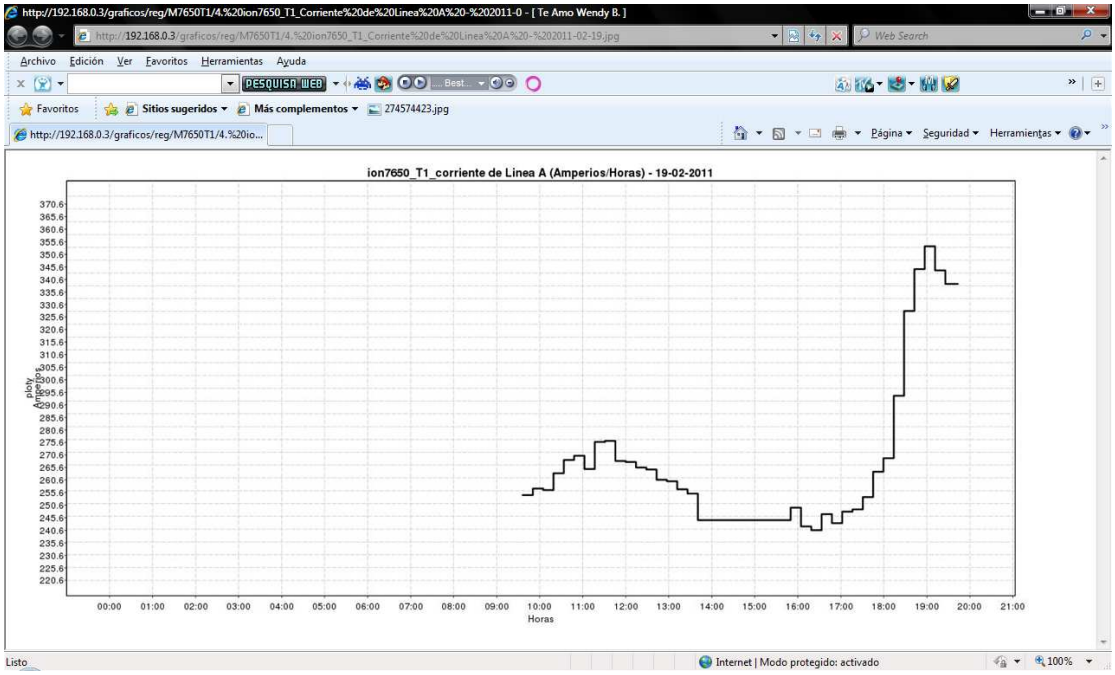


Figura VI.24. Gráfica generada por el sistema en formato jpg.

6.2.4 Centro de monitoreo

Cada central dependiendo de su capacidad de generación y la arquitectura de gestión interna, dispondrá de una o varias estaciones de trabajo e ingeniería ubicadas en los centro de monitoreo.

CAPÍTULO VII

PRUEBA DE LA HIPÓTESIS

7.1 Planteamiento de la hipótesis

La implementación de un Sistema de Monitorización SCADA a través de una solución de software libre para relés y medidores de la EERSA, permitirá la supervisión del funcionamiento de los equipos y además optimizar costos para la implementación de este tipo de sistemas.

7.1.2 Determinación de Causa y Efecto

Causa.- Implementación de un sistema de monitoreo SCADA

Efecto.- Supervisión del buen funcionamiento de los equipos y optimización de costos en la implementación del sistema SCADA.

7.1.3 Selección de variables

Variable dependiente (y)

Garantizar el buen funcionamiento de los equipos, costos en la implementación.

Variable independiente, discreta, cuantitativa (x)

Análisis en interpretación de los datos.

Variables intervinientes w, z

Funcionalidad del sistema y número de datos mostrados.

7.1.4 Análisis de Hipótesis Estadística Operacional

Ho.- La implementación de un Sistema de Monitoreo SCADA a través de una solución de software libre para relés y medidores de la EERSA, permitirá la supervisión del buen funcionamiento de los equipos y optimizar costos en la implementación.

Hi.- La implementación de un Sistema de Monitoreo SCADA a través de una solución de software libre para relés y medidores de la EERSA, no permitirá la supervisión del buen funcionamiento de los equipos y optimizar costos en la implementación.

Ho= Hipótesis Nula

Hi= Hipótesis Alternativa

Ho; u= 7967.43 V (implícito)

H1; u≠ 7967.43 V

Donde el valor de u es el voltaje línea neutro nominal.

Prueba de Hipótesis de media con muestra >30

Contraste de la hipótesis

Con 90% de Aceptación

7.1.4.1 Nivel de Significancia

Para el análisis de nivel de significancia podemos tomarlo a α entre un valor de 0.05 a 0.1 es decir entre un 5 o 10%, en este tema de tesis el valor para α es el siguiente

α=0.1 que es el máximo de 10%

7.1.5 Cálculo del estadístico Z

Para el cálculo del estadístico Z se obtuvo una muestra de 40 datos la cual se presenta en la siguiente tabla.

Tabla VII.XVI. Muestra de datos.

Fecha	Hora	M7330AL1S1_Vlna_(V)
19/02/2011	9:30:00	7931
19/02/2011	9:45:01	7954
19/02/2011	10:00:01	7909
19/02/2011	10:15:00	7948
19/02/2011	10:30:01	7910
19/02/2011	10:45:00	7937
19/02/2011	11:00:01	7928
19/02/2011	11:15:01	7944
19/02/2011	11:30:00	7920
19/02/2011	11:45:00	7903
19/02/2011	12:00:00	7956
19/02/2011	12:15:01	7951
19/02/2011	12:30:00	7972
19/02/2011	12:45:00	8031
19/02/2011	13:00:00	8016
19/02/2011	13:15:00	8073
19/02/2011	13:30:00	8034
19/02/2011	13:45:00	8027

Tabla VII.XVI. Muestra de datos. (Continuación).

19/02/2011	15:45:01	7847
19/02/2011	16:00:01	7956
19/02/2011	16:15:00	7945
19/02/2011	16:30:01	7917
19/02/2011	16:45:00	7904
19/02/2011	17:00:01	7884
19/02/2011	17:15:00	7860
19/02/2011	17:30:01	7971
19/02/2011	17:45:01	7987
19/02/2011	18:00:01	7966
19/02/2011	18:15:00	7996
19/02/2011	18:30:00	7922
19/02/2011	18:45:00	7912
19/02/2011	19:00:01	7907
19/02/2011	19:15:01	7891
19/02/2011	19:30:00	7890
19/02/2011	19:45:01	7903
19/02/2011	20:00:00	7899
19/02/2011	20:15:00	8001
20/02/2011	20:30:00	7956
21/02/2011	20:45:00	7915
22/02/2011	21:00:00	7989

Tabla VII.XVII. Distribución de frecuencias Media muestral y Varianza para valores no agrupados.

Intervalo (V)	**Marca de clase**	**Observaciones F. Absoluta**	**Frec. Relativas**	**Media Muestral**	**Varianza**
Xi	$\frac{(xi+xj)}{2}$	Ni	**fi**=ni/N	$\bar{x}i$ =xi.fi	$\theta i^2=\frac{(xi-\bar{x}i)^2 ni}{N}$
7569.06 – 7648.735	7608.898	0	0	0	0
7648.736 – 7728.41	7688.573	0	0	0	0

Tabla VII.XVII. Distribución de frecuencias Media muestral y Varianza para valores no agrupados. (Continuación).

7728.42 – 7808.085	7768.253	0	0	0	0
7808.08 – 7887.76	7847.92	3	0.075	588.594	3962230.524
7887.77 – 7967.435	7927.603	26	0.65	5152.942	8417859.734
7967.436 – 8047.11	8007.273	10	0.25	2001.818	8836541.007
8047.12 – 8126.785	8086.953	1	0.025	202.174	8566296.597
8126.786 – 8206.46	8166.623	0	0	0	0
8206.47 – 8286.135	8246.303	0	0	0	0
8286.136 – 8365.81	8325.973	0	0	0	0
		$\sum_i ni = N = 40$	$\sum_i fi = 1$	$\bar{x}$ = 7945.528	$\theta^2 = 41707466.086$ 29782927.86

$\theta_{\bar{x}} = \frac{\theta}{\sqrt{n}}$ **;** Desviación Estándar Tipificada

$\mathbf{z} = \frac{\bar{x} - u}{\theta_{\bar{x}}}$ **;**

u ; consideración de la hipótesis nula $\bar{x}$ **;** media muestral

n ; número de elementos muestreados θ **;** Desviación estándar muestral

$\mathbf{z} = \frac{7151.639 - 7736.493}{863.887}$

$\theta_{\bar{x}}$=862.887

$\mathbf{z} = \frac{7151.639 - 7736.493}{863.887}$

z = -0.025

7.1.6 Prueba de una Cola Distribución Normal

Para un nivel de confianza de 10%, según valores ya establecidos

Para α=0,1 z= 1.64

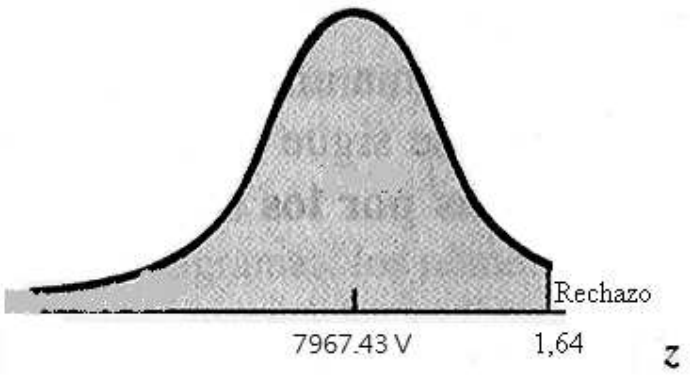


Figura VII.25. Distribución Normal Demostración de hipótesis.

Condición: Rechazar Ho si z> 1,6 sabiendo que 7967.43 V = 0

Conclusión: Ho SE ACEPTA

7.2 Análisis subjetivo de la hipótesis

Costo

Licencia Software ION Enterprise para los 7 medidores de la S/E 1 de la EERSA es de $8820

El software Likindoy – SCADA no requiere de licencias para su funcionamiento.

Ambos software requieren del personal capacitado para su implementación e instalación por lo cual con la solución planteada la EERSA optimizaría el costo en la adquisición de licencias para el software.

Tabla VII.XVIII. Comparación de funciones entre el Software ION Enterprise y el Likindoy – SCADA.

	ION Enterprise	Likindoy – SCADA
Visualización de datos en tiempo real	Si	Si
Cantidad de datos que presenta (Interface HMI)	100%	70%
Configuración de los Puntos de Ajuste (Setpoints).	Si	Si
Autenticación en los Puntos de Ajuste (Setpoints)	Si	Si
Reportes	Si	Si
# de medidores soportados	N	N
Escalabilidad	Si	Si
# de usuarios simultáneos	N	N
Tiempos de respuesta	2 a 3 seg.	1 a 2 seg.
Alarmas	Si	Si
Performance	100%	100%
Generación de Gráficas	Si	Si
Servidor Web	Si	Si
Almacenamiento de Datos	Si	Si
Código Abierto	No	Si

De acuerdo a lo expuesto se puede concluir que el software Likindoy – SCADA presenta el 90% de las funcionalidades del software original ION Enterprise.

Con esto se puede concluir que bajo plataforma Linux es posible diseñar e implementar un sistema de monitoreo para los relés y medidores de la S/E1 con las mismas funcionalidades de la de un software propietario como es el ION Enterprise.

CONCLUSIONES

1. En el proyecto se han analizado y evaluado los resultados obtenidos. Dichos análisis constituyen el fundamento para diseñar e implementar un sistema SCADA para el monitoreo de relés y medidores de la S/E 1 de la EERSA a través de software libre SCADA.

2. Se ha estudiado, comprendido y analizado los parámetros requeridos para la supervisión y el monitoreo optimo de los relés y medidores de la S/E 1 de la EERSA.

3. Se ha estudiado y analizado las conexiones de los dispositivos necesarios que se requieren para el monitoreo de los relés y medidores de la S/E 1 de la EERSA, para lo cual la topología de red existente en la empresa es la adecuada para la implementación del sistema SCADA.

4. En base a las características de las soluciones de software SCADA que se analizaron entre todas las soluciones de software OpenSource existentes en el mercado, se concluyó que el sistema se implementará con Likindoy.

5. La forma más adecuada y efectiva para la transmisión y recepción de datos que se requiere para el monitoreo de los relés y medidores es a través del protocolo de red Modbus TCP/IP, quien se encarga de establecer la comunicación entre los dispositivos de monitoreo con las estaciones de administración.

6. La generación y el diseño de los reportes se ha realizado de acuerdo a los requerimientos del supervisor de la S/E 1 de la EERSA.

RECOMENDACIONES

1. Se recomienda capacitar al personal de operación de la S/E 1 de la EERSA, en cuanto a la interacción con el sistema, ya que esto ayudará a utilizar con eficiencia y eficacia el sistema implementado.

2. Antes de realizar la implementación del sistema es recomendable tener conocimientos sólidos sobre sistemas operativos Linux, programación en Python, sistemas SCADA y el funcionamiento de los relés y medidores en cada uno de los alimentadores de la S/E 1 de la EERSA.

3. La metodología planteada es recomendable para su aplicación en otros alimentadores de la EERSA que actualmente presentan similares características de operación.

4. Se recomienda para la configuración del sistema asegurarse de usar mayúsculas y minúsculas adecuadamente en todo momento ya que el sistema es sensible a estas y considera cadenas distintas.

5. Es recomendable que, para los medidores ION 7330, haya una sola conexión Modbus TCP/IP, ya que éstos no admiten más conexiones simultaneas.

RESUMEN

En el presente estudio se analizo la factibilidad técnica de diseñar e implementar un sistema de monitoreo para relés y medidores de la Sub Estación 1 (S/E 1) de la Empresa Eléctrica Riobamba S.A. (EERSA) a través de software OpenSource SCADA (Supervisory Control And Data Adquisition).

Se realizó un análisis comparativo de alternativas de software de monitoreo entre el OpenSCADA y Likindoy, seleccionando Likindoy por sus grandes características y ventajas tales como el tipo de aplicaciones, gráficas de proceso, visualización de datos a tiempo real, fácil utilización del lenguaje de programación Python, razones por las que se lo utilizó para la creación del sistema de monitoreo y adquisición de datos, implementándolo bajo plataforma Linux Debían Lenny 5.0 y, para la comunicación entre la interface HMI (Human Machine Interface) y el dispositivo, se utilizó protocolo Modbus TCP/IP.

El sistema desarrollado adquiere los registros de voltaje, corriente y potencia a través de red Fast Ethernet y los muestra en la interface HMI que se diseñó con programación php, de tal manera que el operador observa lo que sucede en cada uno de los medidores de cada alimentador, permitiendo además elaborar reportes y gráficas de voltajes, corrientes y potencias en un rango de tiempo predefinido.

Para que el sistema implementado funcione correctamente, es recomendable que, para los medidores ION 7330, haya una sola conexión Modbus TCP/IP, ya que éstos no admiten más conexiones.

SUMMARY

The technical feasibility of designing and implementing a monitoring system for the relays and gauges of the Sub-Station 1 (S/El) of the Electric Enterprise Riobamba S.A (EERSA) was analyzed through OpenSource SCADA software (Supervisory Control and Data Acquisition). A comparative analysis of monitoring software alternatives betweefl th' openSCADA and Likindoy was carried out selecting Likindoy for its great characteristics and advantages such as the type of applications, process graphs, data display at a real time and easy use of the Phyton programming language; these are the reasons why it was used to create a monitoring system and data acquisition implementing it under the Linux Debian Lenny 5.0 and for communication between the interface HMI (Human Machine Interface) and the device. The protocol Modbus TCP/IP was used. The developed system acquires voltage records, current and power through the Fast Ethernet network and shows them in the HMI interface which was designed with the php programming so that the operator observes what is happening in each gauge of each feeder, permitting ,moreover ,to elaborate reports and voltage graphs, currents and powers in a pre-defines time range. To make the implemented system function,it is recommended , for the gauges ION 7330, to have only one connection Modbus TCP/IP as more connections are not admitted.

ANEXOS

Anexo 1. Archivo de Configuración doc/install.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#                        INSTALADOR                        #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani               #
# Fecha: Riobamba, 20 de Mayo del 2010                     #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa        #
# VersiÃ³n: 0000000001                                    #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com      #
############################################################

# Libraries
import curses
import imp

import time

class INSTALLER:

      # Create the object
      def __init__(self):
            # Startup ncurses
            self.__screen=curses.initscr()
            # Start colors
            curses.start_color()
            curses.init_pair(1, curses.COLOR_WHITE, curses.COLOR_BLUE)
            curses.init_pair(2, curses.COLOR_YELLOW,
curses.COLOR_BLACK)
            curses.init_pair(3, curses.COLOR_WHITE, curses.COLOR_RED)
            curses.init_pair(4, curses.COLOR_GREEN, curses.COLOR_BLACK)
            curses.init_pair(5, curses.COLOR_RED, curses.COLOR_BLACK)
            # Show keys strikes on screen
            curses.noecho()
            # One key and action
            curses.cbreak()
            # Percents
            self.__percent_config=0
            self.__percent_process=0
            self.__percent = curses.newwin(1, 100, curses.LINES-3, 4)

      # Destroy the object
      def close(self):
            # Get library once more
            #import curses
            # End the curses application
            curses.nocbreak()
            self.__screen.keypad(0)
            curses.echo()
            # Close the window
            curses.endwin()

      # Print a line
```

```
      def line(self,window, y, text, color):
            window.addstr(y,0,text,color)
            blank=""
            for i in range(len(text),curses.COLS):
                  blank="%s " % (blank)
            window.addstr(y,len(text),blank,color)

      # Print the screen
      def screen(self):
            # Get the screen
            screen=self.__screen
            # Clear
            screen.clear()
            # Head
            self.line(screen,0,"Likindoy installer",
curses.color_pair(1) )
            screen.refresh()
            # Body
            for i in range(1,curses.LINES-1):
                  self.line(screen,i,"---", curses.color_pair(2))
                  pass
            # Tail
            self.line(screen,curses.LINES-2,"Centrologic Computational
Logistic Center - http://www.centrologic.com", curses.color_pair(1) )
            screen.refresh()

      # Percents
      def percent(self,config=None,process=None):
            if (config!=None):
                   self.__percent_config=config
            else:
                  config=self.__percent_config
            if (process!=None):
                   self.__percent_process=process
            else:
                  process=self.__percent_process
            # Config
            config_line=""
            value=config*20/100
            for i in range(0,20):
                  if (i<value):
                        config_line="%s#" % (config_line)
                  else:
                        config_line="%s " % (config_line)
            # Process
            process_line=""
            value=process*20/100
            for i in range(0,20):
                  if (i<value):
                        process_line="%s#" % (process_line)
                  else:
                        process_line="%s " % (process_line)
            # Paint the bar
            self.__percent.addstr(0,0,"[ Config: %s %3d%% ] [ Process:
%s %3d%% ]" % (config_line,config,process_line,process))
            self.__percent.refresh()

      # Main
      def main(self):
```

```
            # Prepare the screen
            self.screen()
            self.percent(config=0,process=0)
            # Show the window for packages
            if (self.packages()):
                  # Show the window for database
                  self.database()

      def test_package(self,win,line,name,package,missing):
            test=False
            try:
                  win.refresh()
                  imp.find_module(name)
                  win.addstr(line,0,"OK",curses.color_pair(4))
                  test=True
            except:
                  win.addstr(line,0,"NO",curses.color_pair(5))
                  missing.append(package)
            win.addstr(line,3,name,curses.color_pair(2))
            win.refresh()
            return test
      def packages(self):
            # Packages
            packages=[]
            packages.append(("MySQLdb","python-mysqldb"))
            packages.append(("rpy","python-rpy"))
            packages.append(("GnuPGInterface","python-gnupginterface"))
            packages.append(("Image","python-imaging"))
            packages.append(("sqlite","python-sqlite"))
            packages.append(("paramiko","python-paramiko"))
            packages.append(("xml","python-xml"))

            # Show percent
            self.percent(config=100,process=0)
            # Start a window
            (begin_x,begin_y,height,width) = (55,2,19,25)
            win = curses.newwin(height, width, begin_y, begin_x)

            # Add text
            all=True
            total=len(packages)
            missing=[]
            win.addstr(0,0," Package test           ",curses.A_REVERSE)
            counter=1
            for (module,debian) in packages:

      all=self.test_package(win,counter,module,debian,missing) and all
                  self.percent(process=counter*100/total)
                  counter+=1

            # List of missing packages
            i=total+1
            if (not all):
                  win.addstr(i+1,0," Missing packages:
",curses.A_REVERSE)
                  i+=2
                  for package in missing:
                        win.addstr(i,0,"%s" % (package))
                        i+=1
```

```
                win.addstr(i+1,0,"- Push a key to finish -")
                win.refresh()
                win.getch()
          return all

    # Database configuration
    def database(self):
          # Import MySQLDB
          import MySQLdb
          # Show percent
          self.percent(config=0,process=0)
          # Start a window
          (begin_x,begin_y,height,width) = (4,2,19,50)
          win = curses.newwin(height, width, begin_y, begin_x)
          # Ask if create database
          create_database=False
          win.addstr(0,0," Create database? (Y/n)",curses.A_REVERSE)
          win.refresh()
          while 1:
                c = win.getch()
                if ((c == ord('y')) or (c == ord('Y')) or (int(c) ==
10)):
                      create_database=True
                      break
                elif ((c == ord('n')) or (c == ord('N'))):
                      create_database=False
                      break

          # Ask for information of database
          if (create_database):
                fine=False
                default_host="127.0.0.1"
                default_port="3306"
                default_socket="auto"
                default_user="root"
                while (not fine):
                      self.percent(config=0)
                      win.addstr( 0,0," Database information (needed
to create database) ",curses.A_REVERSE)
                      win.addstr( 1,0,"Host
")
                      win.addstr( 1,0,"Host (%s):" % (default_host))
                      win.addstr( 2,0,"Port
")
                      win.addstr( 2,0,"Port (%s):" % (default_port))
                      win.addstr( 3,0,"Socket
")
                      win.addstr( 3,0,"Socket (%s):" %
(default_socket))
                      win.addstr( 4,0,"User
")
                      win.addstr( 4,0,"User (%s):" % (default_user))
                      win.addstr( 5,0,"Password:
")
                      win.addstr( 7,0,"
")
                      win.addstr( 8,0,"
")
```

```
                        win.addstr( 9,0,"
")
                        win.addstr(10,0,"
")
                        win.addstr(11,0,"
")
                        win.addstr(12,0,"
")
                        win.addstr(13,0,"
")
                        win.addstr(14,0,"
")
                        win.addstr(15,0,"
")
                        win.addstr(16,0,"
")
                        curses.echo()
                        win.addstr( 1,0,"> Host (%s): " %
(default_host))
                        host=win.getstr()
                        if (host==""):
                              host=default_host
                        else:
                              default_host=host
                        win.addstr( 1,0,"Host:
")
                        win.addstr( 1,0,"Host: %s" % (host))
                        self.percent(config=20)
                        win.addstr( 2,0,"> Port (%s): " %
(default_port))
                        port=win.getstr()
                        if (port==""):
                              port=default_port
                        else:
                              default_port=port
                        win.addstr( 2,0,"Port:
")
                        win.addstr( 2,0,"Port: %s" % (port))
                        self.percent(config=40)
                        win.addstr( 3,0,"> Socket (%s): " %
(default_socket))
                        socket=win.getstr()
                        if (socket==""):
                              socket=default_socket
                        else:
                              default_socket=socket
                        win.addstr( 3,0,"Socket:
")
                        win.addstr( 3,0,"Socket: %s" % (socket))
                        self.percent(config=60)
                        win.addstr( 4,0,"> User (%s): " %
(default_user))
                        user=win.getstr()
                        if (user==""):
                              user=default_user
                        else:
                              default_user=user
                        win.addstr( 4,0,"User:
")
```

```
                        win.addstr( 4,0,"User: %s" % (user))
                        curses.noecho()
                        self.percent(config=80)
                        win.addstr( 5,0,"> Password: ")
                        password=win.getstr()
                        win.addstr( 5,0,"Password: <HIDDEN>
")
                        self.percent(config=100)

                        create_database=False
                        win.addstr(7,0," Is correct the information?
(y/N)",curses.A_REVERSE)
                        win.refresh()
                        while 1:
                              c = win.getch()
                              if ((c == ord('y')) or (c == ord('Y'))):
                                    win.addstr(7,0, " Creating
database...                              ",curses.A_REVERSE)
                                    win.refresh()
                                    try:
                                          win.addstr(8,0, " Connecting
to database...                           ")
                                          win.refresh()
                                          if (socket=="auto"):
                                                # MySQL default socket
                                                BDconnection =
MySQLdb.connect(host=host,port=int(port),user=user, passwd=password)

      self.__BDconnection=BDconnection
                                          else:
                                                # MySQL defined socket
                                                BDconnection =
MySQLdb.connect(host=host,port=int(port),user=user,
passwd=password,unix_socket=socket)
                                          self.percent(process=12.5)
                                          # Activing cursor
                                          win.addstr(9,0, " Getting
cursor...                          ")
                                          win.refresh()
                                          BD = BDconnection.cursor()
                                          self.percent(process=25)
                                          # Create database
                                          win.addstr(10,0, " Creating
database...                       ")
                                          win.refresh()
                                          try:
                                                BD.execute("USE
likindoy_test")
                                                win.addstr(10,0, "
Creating database...[already exists]     ")
                                                win.addstr(11,0, " > I
am not going to create table datos!  ")
                                                win.addstr(12,0, " > I
am not goint to create table ids!    ")

      self.percent(process=62.5)
                                          except:
                                                BD.execute("create
database likindoy_test")
```

```
                                                BD.execute("USE
likindoy_test")

      self.percent(process=37.5)
                                                # Create tables
                                                table_datos='CREATE
TABLE `datos` (  `id` int(11) NOT NULL,  `server` timestamp NOT NULL
default CURRENT_TIMESTAMP,  `fecha` date NOT NULL,  `hora` time NOT
NULL,  `valord` tinyint(1) default NULL,  `valora` float default NULL,
PRIMARY KEY  (`id`,`fecha`,`hora`)) ENGINE=MyISAM;'
                                                table_ids='CREATE TABLE
`ids` (  `id` int(11) NOT NULL auto_increment,  `md5` varchar(32) NOT
NULL,  PRIMARY KEY  (`id`),  UNIQUE KEY `md5` (`md5`)) ENGINE=MyISAM';
                                                win.addstr(11,0, "
Creating table datos...                  ")
                                                win.refresh()
                                                BD.execute(table_datos)

      self.percent(process=50)
                                                win.addstr(12,0, "
Creating table ids...                    ")
                                                win.refresh()
                                                BD.execute(table_ids)
#########################################################################
                                                nombresid="create table
nombresid(id varchar(4),nombreid varchar(10))"
                                                rangos="create table
rangos(id int(11),valoramin float,valoramax float);"
                                                seguridad="create table
seguridad(usuario varchar(10),clave varchar(10),primary
key(usuario,clave));"
                                                clavedefault="insert
into seguridad(usuario,clave) values ('admin','admin');"
                                                BD.execute(nombresid)
                                                BD.execute(rangos)
                                                BD.execute(seguridad)

      BD.execute(clavedefault)
#########################################################################
########################


      self.percent(process=62.5)
                                          # Create user
                                          win.addstr(13,0, " Creating
user...                          ")
                                          win.refresh()
                                          BD.execute("USE mysql")
                                          BD.execute("SELECT User FROM
`user` WHERE `User`='likindoy' AND `Host`='localhost'")
                                          if (BD.rowcount>0):
                                                win.addstr(13,0, "
Creating user...[already exists]         ")
                                          else:
                                                BD.execute("CREATE USER
'likindoy'@'localhost' IDENTIFIED BY 'likindoyclave'")
                                          self.percent(process=75)
                                          # Granting user
```

```
                                        win.addstr(14,0, " Granting
user...                          ")
                                        win.refresh()
                                        query_grant="GRANT USAGE ON *
. * TO 'likindoy'@'localhost' IDENTIFIED BY 'likindoyclave' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR
0 MAX_USER_CONNECTIONS 0 ;"
                                        BD.execute(query_grant)
                                        query_grant="GRANT SELECT ,
INSERT , UPDATE , DELETE ON `likindoy_test` . * TO
'likindoy'@'localhost';"
                                        BD.execute(query_grant)
                                        self.percent(process=87.5)
                                        # Closing connection
                                        win.addstr(15,0, " Clossing
connection to database...       ")
                                        win.refresh()
                                        BD.close()
                                        BDconnection.close()
                                        self.percent(process=100)
                                        # Say that everything was
fine
                                        win.addstr(17,0, " Database
created, push a key to finish.         ",curses.A_REVERSE)
                                        win.refresh()
                                        c=win.getch()
                                        fine=True
                                    except Exception,e:
                                        win.addstr( 7,0,"
")
                                        win.addstr( 8,0,"
")
                                        win.addstr( 9,0,"
")
                                        win.addstr(10,0,"
")
                                        win.addstr(11,0,"
")
                                        win.addstr(12,0,"
")
                                        win.addstr(13,0,"
")
                                        win.addstr(14,0,"
")
                                        win.addstr(15,0,"
")
                                        win.addstr(16,0,"
")
                                        win.addstr(7,0,"ERROR (push a
key to try again):",curses.color_pair(3))
                                        win.addstr(8,0,"%s" %
(e),curses.color_pair(3))
                                        win.refresh()
                                        fine=False
                                        c = win.getch()
                                    break
                                elif ((c == ord('n')) or (c == ord('N'))
or (int(c) == 10)):
                                    break
```

```
# === MAIN === ================================================
if __name__=='__main__':
      i=INSTALLER()
      try:
            i.main()
            i.close()
      except:
            import traceback
            i.close()
            traceback.print_exc()
```

Anexo 2. Archivo de Configuración /etc/adquirir_momentum.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#              CONFIGURACION DE ADQUISICION                #
############################################################
# Autor: Francisco Narvàez, Wilmer Guamani                 #
# Fecha: Riobamba, 12 de Julio del 2010                    #
# Descripción: Instalador de Scada-Likindoy-Eersa          #
# Versión: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

# Cargo la configuracion por defecto
from adquirir_momentum_default import *
from src.lib.REGISTRADORES import *
from src.lib.FILE_SIGNALS import *
from src.lib.NET_UDP import *
from src.etcbin.senales import *

#PLANTILLA
####CONFIGURACION GENERAL DEL LOS <NOMBRE GENERAL DE EQUIPOS>#########

# Configuracion de <EQUIPO Nº1>
#reg_<EQUIPO Nº1>=[]
#unidades_<EQUIPO Nº1>=[]
#divisor_<EQUIPO Nº1>=[]
#reportes_<EQUIPO Nº1>=[]

#reg_<EQUIPO Nº1>.append("<SEÑAL 1 DE EQUIPO 1>")
#reportes_<EQUIPO Nº1>.append("<SEÑAL 1 DE EQUIPO 1>")
#unidades_<EQUIPO Nº1>.append('<UNIDAD (Ejm: "V"(VOLTIOS))>')
#divisor_<EQUIPO Nº1>.append('<DIVISOR (Ejm: "1")>')

#reg_<EQUIPO Nº1>.append("<SEÑAL 2 DE EQUIPO 1>")
#reportes_<EQUIPO Nº1>.append("<SEÑAL 2 DE EQUIPO 1>")
#unidades_<EQUIPO Nº1>.append('<UNIDAD (Ejm: "Hz"(FRECUENCIA))>')
#divisor_<EQUIPO Nº1>.append('<DIVISOR (Ejm: "10")>')
#.......
#reportes_telemando.append(reportes_<EQUIPO Nº1>)
#num_senales_telemando.append(len(reg_<EQUIPO Nº1>))
#unidades_telemando.append(unidades_<EQUIPO Nº1>)
#divisor_telemando.append(divisor_<EQUIPO Nº1>)

# Configuracion de <EQUIPO Nº2>
#reg_<EQUIPO Nº2>=[]
#unidades_<EQUIPO Nº2>=[]
#divisor_<EQUIPO Nº2>=[]
#reportes_<EQUIPO Nº2>=[]

#reg_<EQUIPO Nº2>.append("<SEÑAL 1 DE EQUIPO 2>")
#reportes_<EQUIPO Nº2>.append("<SEÑAL 1 DE EQUIPO 2>")
#unidades_<EQUIPO Nº2>.append('<UNIDAD (Ejm: "A"(AMPERIOS))>')
#divisor_<EQUIPO Nº2>.append('<DIVISOR (Ejm: "100")>')
```

```
#reg_<EQUIPO Nº2>.append("<SEÑAL 2 DE EQUIPO 2>")
#reportes_<EQUIPO Nº2>.append("<SEÑAL 2 DE EQUIPO 2>")
#unidades_<EQUIPO Nº2>.append('<UNIDAD (Ejm: "KW"(PÒTENCIA))>')
#divisor_<EQUIPO Nº2>.append('<DIVISOR (Ejm: "1")>')
#.......
#reportes_telemando.append(reportes_<EQUIPO Nº2>)
#num_senales_telemando.append(len(reg_<EQUIPO Nº2>))
#unidades_telemando.append(unidades_<EQUIPO Nº2>)
#divisor_telemando.append(divisor_<EQUIPO Nº2>)

############################CONFIGURACION DE
REGISTRADORES.############################

# Registrador del <EQUIPO Nº1>
#registradores<EQUIPO
Nº1>=REGISTRADORES([FILE_SIGNALS("momentum","data/reg/<EQUIPO
Nº1>_",compress=True)])

# Registrador del <EQUIPO Nº2>
#registradores<EQUIPO
Nº2>=REGISTRADORES([FILE_SIGNALS("momentum","data/reg/<EQUIPO
Nº2>_",compress=True)])

############################CONFIGURACION DE LOS
TELEMANDOS############################

# Creo el Telemando para <EQUIPO Nº1>
#<EQUIPO Nº1>=TELEMANDO("<EQUIPO Nº1>",registradores<EQUIPO Nº1>,s)
#<EQUIPO Nº1>.set_debug(True)
#<EQUIPO Nº1>.config(config,"<DIRECCION IP DEL EQUIPO Nº1>")
#<EQUIPO Nº1>.load(reg_<EQUIPO Nº1>)
#nombre_telemandos.append('<EQUIPO Nº1>')
#listado_telemandos.append(<EQUIPO Nº1>)

# === GENERAL ===

############################CONFIGURACION GENERAL DE LOS
MEDIDORES########################


# Configuracion de medidor 7650 Transformador 1
reg_M7650_T1=[]
unidades_M7650_T1=[]
divisor_M7650_T1=[]
reportes_M7650_T1=[]

reg_M7650_T1.append("M7650T1_Vlna")
reportes_M7650_T1.append("M7650T1_Vlna")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vlnb")
reportes_M7650_T1.append("M7650T1_Vlnb")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vlnc")
reportes_M7650_T1.append("M7650T1_Vlnc")
unidades_M7650_T1.append('V')
```

```
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vlnavg")
reportes_M7650_T1.append("M7650T1_Vlnavg")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vllab")
reportes_M7650_T1.append("M7650T1_Vllab")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vllbc")
reportes_M7650_T1.append("M7650T1_Vllbc")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vllca")
reportes_M7650_T1.append("M7650T1_Vllca")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Vllavg")
reportes_M7650_T1.append("M7650T1_Vllavg")
unidades_M7650_T1.append('V')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_Ia")
reportes_M7650_T1.append("M7650T1_Ia")
unidades_M7650_T1.append('A')
divisor_M7650_T1.append('10')

reg_M7650_T1.append("M7650T1_Ib")
reportes_M7650_T1.append("M7650T1_Ib")
unidades_M7650_T1.append('A')
divisor_M7650_T1.append('10')

reg_M7650_T1.append("M7650T1_Ic")
reportes_M7650_T1.append("M7650T1_Ic")
unidades_M7650_T1.append('A')
divisor_M7650_T1.append('10')

reg_M7650_T1.append("M7650T1_Iavg")
reportes_M7650_T1.append("M7650T1_Iavg")
unidades_M7650_T1.append('A')
divisor_M7650_T1.append('10')

reg_M7650_T1.append("M7650T1_Freq")
reportes_M7650_T1.append("M7650T1_Freq")
unidades_M7650_T1.append('Hz')
divisor_M7650_T1.append('10')

reg_M7650_T1.append("M7650T1_KWa")
reportes_M7650_T1.append("M7650T1_KWa")
unidades_M7650_T1.append('KW')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_KWb")
reportes_M7650_T1.append("M7650T1_KWb")
```

```
unidades_M7650_T1.append('KW')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_KWc")
reportes_M7650_T1.append("M7650T1_KWc")
unidades_M7650_T1.append('KW')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_KWtot")
reportes_M7650_T1.append("M7650T1_KWtot")
unidades_M7650_T1.append('KW')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_KVARtot")
reportes_M7650_T1.append("M7650T1_KVARtot")
unidades_M7650_T1.append('KVAR')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_KVAtot")
reportes_M7650_T1.append("M7650T1_KVAtot")
unidades_M7650_T1.append('KVA')
divisor_M7650_T1.append('1')

reg_M7650_T1.append("M7650T1_PFsigntot")
#reportes_M7650_T1.append("M7650T1_PFsigntot")
unidades_M7650_T1.append('%')
divisor_M7650_T1.append('100')

reportes_telemando.append(reportes_M7650_T1)
num_senales_telemando.append(len(reg_M7650_T1))
unidades_telemando.append(unidades_M7650_T1)
divisor_telemando.append(divisor_M7650_T1)

#Configuracion Ion 7330 Alimentedor 1 Subestacion 1
reg_M7330_AL_1_1=[]
unidades_M7330_AL_1_1=[]
divisor_M7330_AL_1_1=[]
reportes_M7330_AL_1_1=[]

reg_M7330_AL_1_1.append("M7330AL1S1_Vlna")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vlna")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Vlnb")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vlnb")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Vlnc")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vlnc")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Vlnavg")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vlnavg")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')
```

```
reg_M7330_AL_1_1.append("M7330AL1S1_Vllab")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vllab")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Vllbc")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vllbc")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Vllca")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vllca")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Vllavg")
reportes_M7330_AL_1_1.append("M7330AL1S1_Vllavg")
unidades_M7330_AL_1_1.append('V')
divisor_M7330_AL_1_1.append('1')

reg_M7330_AL_1_1.append("M7330AL1S1_Ia")
reportes_M7330_AL_1_1.append("M7330AL1S1_Ia")
unidades_M7330_AL_1_1.append('A')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_Ib")
reportes_M7330_AL_1_1.append("M7330AL1S1_Ib")
unidades_M7330_AL_1_1.append('A')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_Ic")
reportes_M7330_AL_1_1.append("M7330AL1S1_Ic")
unidades_M7330_AL_1_1.append('A')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_Iavg")
reportes_M7330_AL_1_1.append("M7330AL1S1_Iavg")
unidades_M7330_AL_1_1.append('A')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_Freq")
reportes_M7330_AL_1_1.append("M7330AL1S1_Freq")
unidades_M7330_AL_1_1.append('Hz')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_KWa")
reportes_M7330_AL_1_1.append("M7330AL1S1_KWa")
unidades_M7330_AL_1_1.append('KW')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_KWb")
reportes_M7330_AL_1_1.append("M7330AL1S1_KWb")
unidades_M7330_AL_1_1.append('KW')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_KWc")
reportes_M7330_AL_1_1.append("M7330AL1S1_KWc")
unidades_M7330_AL_1_1.append('KW')
divisor_M7330_AL_1_1.append('10')
```

```
reg_M7330_AL_1_1.append("M7330AL1S1_KWtot")
reportes_M7330_AL_1_1.append("M7330AL1S1_KWtot")
unidades_M7330_AL_1_1.append('KW')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_KVARtot")
reportes_M7330_AL_1_1.append("M7330AL1S1_KVARtot")
unidades_M7330_AL_1_1.append('KVAR')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_KVAtot")
reportes_M7330_AL_1_1.append("M7330AL1S1_KVAtot")
unidades_M7330_AL_1_1.append('KVA')
divisor_M7330_AL_1_1.append('10')

reg_M7330_AL_1_1.append("M7330AL1S1_PFsigntot")
reportes_M7330_AL_1_1.append("M7330AL1S1_PFsigntot")
unidades_M7330_AL_1_1.append('%')
divisor_M7330_AL_1_1.append('10')

reportes_telemando.append(reportes_M7330_AL_1_1)
num_senales_telemando.append(len(reg_M7330_AL_1_1))
unidades_telemando.append(unidades_M7330_AL_1_1)
divisor_telemando.append(divisor_M7330_AL_1_1)

#Configuracion Ion 7330 Alimentedor 2 Subestacion 1
reg_M7330_AL_2_1=[]
unidades_M7330_AL_2_1=[]
divisor_M7330_AL_2_1=[]
reportes_M7330_AL_2_1=[]

reg_M7330_AL_2_1.append("M7330AL2S1_Vlna")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vlna")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vlnb")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vlnb")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vlnc")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vlnc")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vlnavg")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vlnavg")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vllab")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vllab")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vllbc")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vllbc")
unidades_M7330_AL_2_1.append('V')
```

```
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vllca")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vllca")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Vllavg")
reportes_M7330_AL_2_1.append("M7330AL2S1_Vllavg")
unidades_M7330_AL_2_1.append('V')
divisor_M7330_AL_2_1.append('1')

reg_M7330_AL_2_1.append("M7330AL2S1_Ia")
reportes_M7330_AL_2_1.append("M7330AL2S1_Ia")
unidades_M7330_AL_2_1.append('A')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_Ib")
reportes_M7330_AL_2_1.append("M7330AL2S1_Ib")
unidades_M7330_AL_2_1.append('A')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_Ic")
reportes_M7330_AL_2_1.append("M7330AL2S1_Ic")
unidades_M7330_AL_2_1.append('A')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_Iavg")
reportes_M7330_AL_2_1.append("M7330AL2S1_Iavg")
unidades_M7330_AL_2_1.append('A')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_Freq")
reportes_M7330_AL_2_1.append("M7330AL2S1_Freq")
unidades_M7330_AL_2_1.append('Hz')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_KWa")
reportes_M7330_AL_2_1.append("M7330AL2S1_KWa")
unidades_M7330_AL_2_1.append('KW')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_KWb")
reportes_M7330_AL_2_1.append("M7330AL2S1_KWb")
unidades_M7330_AL_2_1.append('KW')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_KWc")
reportes_M7330_AL_2_1.append("M7330AL2S1_KWc")
unidades_M7330_AL_2_1.append('KW')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_KWtot")
reportes_M7330_AL_2_1.append("M7330AL2S1_KWtot")
unidades_M7330_AL_2_1.append('KW')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_KVARtot")
reportes_M7330_AL_2_1.append("M7330AL2S1_KVARtot")
```

```
unidades_M7330_AL_2_1.append('KVAR')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_KVAtot")
reportes_M7330_AL_2_1.append("M7330AL2S1_KVAtot")
unidades_M7330_AL_2_1.append('KVA')
divisor_M7330_AL_2_1.append('10')

reg_M7330_AL_2_1.append("M7330AL2S1_PFsigntot")
reportes_M7330_AL_2_1.append("M7330AL2S1_PFsigntot")
unidades_M7330_AL_2_1.append('%')
divisor_M7330_AL_2_1.append('10')

reportes_telemando.append(reportes_M7330_AL_2_1)
num_senales_telemando.append(len(reg_M7330_AL_2_1))
unidades_telemando.append(unidades_M7330_AL_2_1)
divisor_telemando.append(divisor_M7330_AL_2_1)

#Configuracion Ion 7330 Alimentedor 3 Subestacion 1
reg_M7330_AL_3_1=[]
unidades_M7330_AL_3_1=[]
divisor_M7330_AL_3_1=[]
reportes_M7330_AL_3_1=[]

reg_M7330_AL_3_1.append("M7330AL3S1_Vlna")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vlna")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Vlnb")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vlnb")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Vlnc")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vlnc")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Vlnavg")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vlnavg")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Vllab")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vllab")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Vllbc")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vllbc")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Vllca")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vllca")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')
```

```
reg_M7330_AL_3_1.append("M7330AL3S1_Vllavg")
reportes_M7330_AL_3_1.append("M7330AL3S1_Vllavg")
unidades_M7330_AL_3_1.append('V')
divisor_M7330_AL_3_1.append('1')

reg_M7330_AL_3_1.append("M7330AL3S1_Ia")
reportes_M7330_AL_3_1.append("M7330AL3S1_Ia")
unidades_M7330_AL_3_1.append('A')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_Ib")
reportes_M7330_AL_3_1.append("M7330AL3S1_Ib")
unidades_M7330_AL_3_1.append('A')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_Ic")
reportes_M7330_AL_3_1.append("M7330AL3S1_Ic")
unidades_M7330_AL_3_1.append('A')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_Iavg")
reportes_M7330_AL_3_1.append("M7330AL3S1_Iavg")
unidades_M7330_AL_3_1.append('A')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_Freq")
reportes_M7330_AL_3_1.append("M7330AL3S1_Freq")
unidades_M7330_AL_3_1.append('Hz')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_KWa")
reportes_M7330_AL_3_1.append("M7330AL3S1_KWa")
unidades_M7330_AL_3_1.append('KW')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_KWb")
reportes_M7330_AL_3_1.append("M7330AL3S1_KWb")
unidades_M7330_AL_3_1.append('KW')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_KWc")
reportes_M7330_AL_3_1.append("M7330AL3S1_KWc")
unidades_M7330_AL_3_1.append('KW')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_KWtot")
reportes_M7330_AL_3_1.append("M7330AL3S1_KWtot")
unidades_M7330_AL_3_1.append('KW')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_KVARtot")
reportes_M7330_AL_3_1.append("M7330AL3S1_KVARtot")
unidades_M7330_AL_3_1.append('KVAR')
divisor_M7330_AL_3_1.append('10')

reg_M7330_AL_3_1.append("M7330AL3S1_KVAtot")
reportes_M7330_AL_3_1.append("M7330AL3S1_KVAtot")
unidades_M7330_AL_3_1.append('KVA')
divisor_M7330_AL_3_1.append('10')
```

```
reg_M7330_AL_3_1.append("M7330AL3S1_PFsigntot")
reportes_M7330_AL_3_1.append("M7330AL3S1_PFsigntot")
unidades_M7330_AL_3_1.append('%')
divisor_M7330_AL_3_1.append('10')

reportes_telemando.append(reportes_M7330_AL_3_1)
num_senales_telemando.append(len(reg_M7330_AL_3_1))
unidades_telemando.append(unidades_M7330_AL_3_1)
divisor_telemando.append(divisor_M7330_AL_3_1)

#Configuracion Ion 7330 Alimentedor 4 Subestacion 1
reg_M7330_AL_4_1=[]
unidades_M7330_AL_4_1=[]
divisor_M7330_AL_4_1=[]
reportes_M7330_AL_4_1=[]

reg_M7330_AL_4_1.append("M7330AL4S1_Vlna")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vlna")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vlnb")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vlnb")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vlnc")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vlnc")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vlnavg")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vlnavg")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vllab")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vllab")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vllbc")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vllbc")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vllca")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vllca")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Vllavg")
reportes_M7330_AL_4_1.append("M7330AL4S1_Vllavg")
unidades_M7330_AL_4_1.append('V')
divisor_M7330_AL_4_1.append('1')

reg_M7330_AL_4_1.append("M7330AL4S1_Ia")
reportes_M7330_AL_4_1.append("M7330AL4S1_Ia")
unidades_M7330_AL_4_1.append('A')
```

```
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_Ib")
reportes_M7330_AL_4_1.append("M7330AL4S1_Ib")
unidades_M7330_AL_4_1.append('A')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_Ic")
reportes_M7330_AL_4_1.append("M7330AL4S1_Ic")
unidades_M7330_AL_4_1.append('A')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_Iavg")
reportes_M7330_AL_4_1.append("M7330AL4S1_Iavg")
unidades_M7330_AL_4_1.append('A')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_Freq")
reportes_M7330_AL_4_1.append("M7330AL4S1_Freq")
unidades_M7330_AL_4_1.append('Hz')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_KWa")
reportes_M7330_AL_4_1.append("M7330AL4S1_KWa")
unidades_M7330_AL_4_1.append('KW')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_KWb")
reportes_M7330_AL_4_1.append("M7330AL4S1_KWb")
unidades_M7330_AL_4_1.append('KW')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_KWc")
reportes_M7330_AL_4_1.append("M7330AL4S1_KWc")
unidades_M7330_AL_4_1.append('KW')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_KWtot")
reportes_M7330_AL_4_1.append("M7330AL4S1_KWtot")
unidades_M7330_AL_4_1.append('KW')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_KVARtot")
reportes_M7330_AL_4_1.append("M7330AL4S1_KVARtot")
unidades_M7330_AL_4_1.append('KVAR')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_KVAtot")
reportes_M7330_AL_4_1.append("M7330AL4S1_KVAtot")
unidades_M7330_AL_4_1.append('KVA')
divisor_M7330_AL_4_1.append('10')

reg_M7330_AL_4_1.append("M7330AL4S1_PFsigntot")
reportes_M7330_AL_4_1.append("M7330AL4S1_PFsigntot")
unidades_M7330_AL_4_1.append('%')
divisor_M7330_AL_4_1.append('10')

reportes_telemando.append(reportes_M7330_AL_4_1)
num_senales_telemando.append(len(reg_M7330_AL_4_1))
```

```
unidades_telemando.append(unidades_M7330_AL_4_1)
divisor_telemando.append(divisor_M7330_AL_4_1)

#Configuracion Ion 7330 Alimentedor 5 Subestacion 1
reg_M7330_AL_5_1=[]
unidades_M7330_AL_5_1=[]
divisor_M7330_AL_5_1=[]
reportes_M7330_AL_5_1=[]

reg_M7330_AL_5_1.append("M7330AL5S1_Vlna")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vlna")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vlnb")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vlnb")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vlnc")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vlnc")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vlnavg")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vlnavg")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vllab")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vllab")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vllbc")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vllbc")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vllca")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vllca")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Vllavg")
reportes_M7330_AL_5_1.append("M7330AL5S1_Vllavg")
unidades_M7330_AL_5_1.append('V')
divisor_M7330_AL_5_1.append('1')

reg_M7330_AL_5_1.append("M7330AL5S1_Ia")
reportes_M7330_AL_5_1.append("M7330AL5S1_Ia")
unidades_M7330_AL_5_1.append('A')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_Ib")
reportes_M7330_AL_5_1.append("M7330AL5S1_Ib")
unidades_M7330_AL_5_1.append('A')
divisor_M7330_AL_5_1.append('10')
```

```
reg_M7330_AL_5_1.append("M7330AL5S1_Ic")
reportes_M7330_AL_5_1.append("M7330AL5S1_Ic")
unidades_M7330_AL_5_1.append('A')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_Iavg")
reportes_M7330_AL_5_1.append("M7330AL5S1_Iavg")
unidades_M7330_AL_5_1.append('A')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_Freq")
reportes_M7330_AL_5_1.append("M7330AL5S1_Freq")
unidades_M7330_AL_5_1.append('Hz')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_KWa")
reportes_M7330_AL_5_1.append("M7330AL5S1_KWa")
unidades_M7330_AL_5_1.append('KW')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_KWb")
reportes_M7330_AL_5_1.append("M7330AL5S1_KWb")
unidades_M7330_AL_5_1.append('KW')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_KWc")
reportes_M7330_AL_5_1.append("M7330AL5S1_KWc")
unidades_M7330_AL_5_1.append('KW')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_KWtot")
reportes_M7330_AL_5_1.append("M7330AL5S1_KWtot")
unidades_M7330_AL_5_1.append('KW')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_KVARtot")
reportes_M7330_AL_5_1.append("M7330AL5S1_KVARtot")
unidades_M7330_AL_5_1.append('KVAR')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_KVAtot")
reportes_M7330_AL_5_1.append("M7330AL5S1_KVAtot")
unidades_M7330_AL_5_1.append('KVA')
divisor_M7330_AL_5_1.append('10')

reg_M7330_AL_5_1.append("M7330AL5S1_PFsigntot")
reportes_M7330_AL_5_1.append("M7330AL5S1_PFsigntot")
unidades_M7330_AL_5_1.append('%')
divisor_M7330_AL_5_1.append('10')

reportes_telemando.append(reportes_M7330_AL_5_1)
num_senales_telemando.append(len(reg_M7330_AL_5_1))
unidades_telemando.append(unidades_M7330_AL_5_1)
divisor_telemando.append(divisor_M7330_AL_5_1)

#Configuracion Ion 7330 Alimentedor 6 Subestacion 1
reg_M7330_AL_6_1=[]
unidades_M7330_AL_6_1=[]
divisor_M7330_AL_6_1=[]
```

```
reportes_M7330_AL_6_1=[]

reg_M7330_AL_6_1.append("M7330AL6S1_Vlna")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vlna")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vlnb")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vlnb")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vlnc")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vlnc")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vlnavg")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vlnavg")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vllab")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vllab")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vllbc")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vllbc")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vllca")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vllca")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Vllavg")
reportes_M7330_AL_6_1.append("M7330AL6S1_Vllavg")
unidades_M7330_AL_6_1.append('V')
divisor_M7330_AL_6_1.append('1')

reg_M7330_AL_6_1.append("M7330AL6S1_Ia")
reportes_M7330_AL_6_1.append("M7330AL6S1_Ia")
unidades_M7330_AL_6_1.append('A')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_Ib")
reportes_M7330_AL_6_1.append("M7330AL6S1_Ib")
unidades_M7330_AL_6_1.append('A')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_Ic")
reportes_M7330_AL_6_1.append("M7330AL6S1_Ic")
unidades_M7330_AL_6_1.append('A')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_Iavg")
reportes_M7330_AL_6_1.append("M7330AL6S1_Iavg")
```

```
unidades_M7330_AL_6_1.append('A')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_Freq")
reportes_M7330_AL_6_1.append("M7330AL6S1_Freq")
unidades_M7330_AL_6_1.append('Hz')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_KWa")
reportes_M7330_AL_6_1.append("M7330AL6S1_KWa")
unidades_M7330_AL_6_1.append('KW')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_KWb")
reportes_M7330_AL_6_1.append("M7330AL6S1_KWb")
unidades_M7330_AL_6_1.append('KW')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_KWc")
reportes_M7330_AL_6_1.append("M7330AL6S1_KWc")
unidades_M7330_AL_6_1.append('KW')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_KWtot")
reportes_M7330_AL_6_1.append("M7330AL6S1_KWtot")
unidades_M7330_AL_6_1.append('KW')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_KVARtot")
reportes_M7330_AL_6_1.append("M7330AL6S1_KVARtot")
unidades_M7330_AL_6_1.append('KVAR')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_KVAtot")
reportes_M7330_AL_6_1.append("M7330AL6S1_KVAtot")
unidades_M7330_AL_6_1.append('KVA')
divisor_M7330_AL_6_1.append('10')

reg_M7330_AL_6_1.append("M7330AL6S1_PFsigntot")
reportes_M7330_AL_6_1.append("M7330AL6S1_PFsigntot")
unidades_M7330_AL_6_1.append('%')
divisor_M7330_AL_6_1.append('10')

reportes_telemando.append(reportes_M7330_AL_6_1)
num_senales_telemando.append(len(reg_M7330_AL_6_1))
unidades_telemando.append(unidades_M7330_AL_6_1)
divisor_telemando.append(divisor_M7330_AL_6_1)

############################CONFIGURACION
REGISTRADORES#################################

# Creo los registradores que vamos a usar

# Registrador del medidor ion 7650 transformador 1
registradoresion7650_T1=REGISTRADORES([FILE_SIGNALS("momentum","data/r
eg/M7650T1_",compress=True)])

# Registrador del medidor ion 7330 alimentador 1 subestacion 1
```

```
registradoresion7330_AL_1_1=REGISTRADORES([FILE_SIGNALS("momentum","da
ta/reg/M7330AL1S1_",compress=True)])


# Registrador del medidor ion 7330 alimentador 2 subestacion 1
registradoresion7330_AL_2_1=REGISTRADORES([FILE_SIGNALS("momentum","da
ta/reg/M7330AL2S1_",compress=True)])


# Registrador del medidor ion 7330 alimentador 3 subestacion 1
registradoresion7330_AL_3_1=REGISTRADORES([FILE_SIGNALS("momentum","da
ta/reg/M7330AL3S1_",compress=True)])


# Registrador del medidor ion 7330 alimentador 4 subestacion 1
registradoresion7330_AL_4_1=REGISTRADORES([FILE_SIGNALS("momentum","da
ta/reg/M7330AL4S1_",compress=True)])


# Registrador del medidor ion 7330 alimentador 5 subestacion 1
registradoresion7330_AL_5_1=REGISTRADORES([FILE_SIGNALS("momentum","da
ta/reg/M7330AL5S1_",compress=True)])


# Registrador del medidor ion 7330 alimentador 6 subestacion 1
registradoresion7330_AL_6_1=REGISTRADORES([FILE_SIGNALS("momentum","da
ta/reg/M7330AL6S1_",compress=True)])


############################CONFIGURACION DE LOS
TELEMANDOS############################

# Creo el ION 7650 TRANSFORMADOR 1
M7650T1=TELEMANDO("M7650_T1",registradoresion7650_T1,s)
M7650T1.set_debug(True)
M7650T1.config(config,"192.168.0.166")
M7650T1.load(reg_M7650_T1)
nombre_telemandos.append('M7650T1')
listado_telemandos.append(M7650T1)


#Creo el ion 7330 Alimentador 1 a Subestacion 1
M7330AL1S1=TELEMANDO("M7330_AL1S1",registradoresion7330_AL_1_1,s)
M7330AL1S1.set_debug(True)
M7330AL1S1.config(config,"192.168.0.220")
M7330AL1S1.load(reg_M7330_AL_1_1)
nombre_telemandos.append('M7330AL1S1')
listado_telemandos.append(M7330AL1S1)


#Creo el ion 7330 Alimentador 2 a Subestacion 1
M7330AL2S1=TELEMANDO("M7330_AL2S1",registradoresion7330_AL_2_1,s)
M7330AL2S1.set_debug(True)
M7330AL2S1.config(config,"192.168.0.226")
M7330AL2S1.load(reg_M7330_AL_2_1)
nombre_telemandos.append('M7330AL2S1')
listado_telemandos.append(M7330AL2S1)
```

```
#Creo el ion 7330 Alimentador 3 a Subestacion 1
M7330AL3S1=TELEMANDO("M7330_AL3S1",registradoresion7330_AL_3_1,s)
M7330AL3S1.set_debug(True)
M7330AL3S1.config(config,"192.168.0.222")
M7330AL3S1.load(reg_M7330_AL_3_1)
nombre_telemandos.append('M7330AL3S1')
listado_telemandos.append(M7330AL3S1)


#Creo el ion 7330 Alimentador 4 a Subestacion 1
M7330AL4S1=TELEMANDO("M7330_AL4S1",registradoresion7330_AL_4_1,s)
M7330AL4S1.set_debug(True)
M7330AL4S1.config(config,"192.168.0.169")
M7330AL4S1.load(reg_M7330_AL_4_1)
nombre_telemandos.append('M7330AL4S1')
listado_telemandos.append(M7330AL4S1)



#Creo el ion 7330 Alimentador 5 a Subestacion 1
M7330AL5S1=TELEMANDO("M7330_AL5S1",registradoresion7330_AL_5_1,s)
M7330AL5S1.set_debug(True)
M7330AL5S1.config(config,"192.168.0.224")
M7330AL5S1.load(reg_M7330_AL_5_1)
nombre_telemandos.append('M7330AL5S1')
listado_telemandos.append(M7330AL5S1)

#Creo el ion 7330 Alimentador 6 a Subestacion 1
M7330AL6S1=TELEMANDO("M7330_AL6S1",registradoresion7330_AL_6_1,s)
M7330AL6S1.set_debug(True)
M7330AL6S1.config(config,"192.168.0.225")
M7330AL6S1.load(reg_M7330_AL_6_1)
nombre_telemandos.append('M7330AL6S1')
listado_telemandos.append(M7330AL6S1)

########################################################################
#########################################################
########################################################################
#########################################################
# Calculo de id por cada telemando, esto se usarà en cualquier
problema de adquisicion, para llenar los campos con null   #
tam=len(num_senales_telemando)
contador_senales=0

for i in num_senales_telemando:
      if (contador_senales == 0):
            num_senales_telemando_ini.append(contador_senales)
            num_senales_telemando_fin.append(i)

      else: #(contador_senales == 1)

      num_senales_telemando_ini.append(num_senales_telemando_fin[conta
dor_senales-1]+1)

      num_senales_telemando_fin.append(num_senales_telemando_fin[conta
dor_senales-1]+i)

      contador_senales=contador_senales+1
#     print "contador=",contador_senales
```

```
#     print "actual=",i
#     print "contador inicio: ",num_senales_telemando_ini , " final:
",num_senales_telemando_fin
#     time.sleep(5)
```

Anexo 3. Archivo de Configuración adquirir_momentum_M7650T1.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS V 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                 #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa          #
# VersiÃ³n: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

#############################################################################
##############################
#################################################(EDITAR)###############
##############################
# Aqui debe declarar el numero de telemando para el cual funcionarÃ
esta libreria adquirir, ya que #
# solo para ese telemando adquirira datos, no olvide que se cuenta en
orden de declaraciÃ²n el en   #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################
##############################
NUM_TELEMANDO=0

#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################

# Librerias {{{1
# Cargo la configuracion por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# DecisiÃ³n segÃºn parÃ¡metros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
```

```
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=unidades_telemando[NUM_TELEMANDO]

      # listado de divisores del telemando actual
divisores_telemando_actual=divisor_telemando[NUM_TELEMANDO]

# Inicializo los datos
vez=1
end=False
escribir=False

# lista de ids_unidades y relacion de ids con registros
ids_telemando=[]
ids_registros=[]
band=False

try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecucciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # control para cargar el dato en la db.
                  guardar=sql.condicion_tiempo()
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
#                       print "entro 1 guardar=",guardar, "band ",
band," ids_registros=",ids_registros, " divisores
telemando=",divisores_telemando_actual
#                       time.sleep(1)
```

```
      lista_retorno=t.write(guardar,band,ids_registros,divisores_telem
ando_actual)
#                       print "lista de retorno....",lista_retorno
#                       time.sleep(5)
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
#                             print "Error en la AdquisiciÃ²n.
Comprueve la conexiÃ²n y vuelva a reiniciar la adquisiciÃ²n."
#                             time.sleep(2)
                        # Obtengo los ids de la seÃ±al
                              # lista en donde obtengo todos los ids.

                        if (band == False):
                              for datos in lista_retorno:
#                                   print "datos...",datos

      ids=sql.ids_unidades(telemando_actual,datos)
                                    ids_telemando.append(ids[0])
#                                   print "lista de ids=",ids_telemando
#                                   time.sleep(5)

      ids_registros.append((ids[0],ids[1]))
#                                   print "lista de registros
ids=",ids_registros
                                    band=True
#                       print "ids telemando=",ids_telemando
#                       time.sleep(2)

                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,diviso
res_telemando_actual,ids_telemando)
#                             print "unidades del
telemando=",unidades_telemando_actual
#                             print "datos actuales: ", datos
#                             time.sleep(2)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores

#                 print "guardar==",guardar
#                 time.sleep(2)
                  if (guardar == True):
#                       print "paso......."
```

```
#                       time.sleep(2)
                        for registrador in listado_registradores:
                              # Otengo el nombre del fichero de salida
                              try:
                                    # Fichero de salida
                                    fichero=registrador.fichero()
                              except Exception:
                                    # No hay fichero de salida en este
registrador
                                    fichero=None
                              # Si existe fichero de salida, lo proceso
                              if (fichero):
                                    # Compruebo la existencia del
fichero FLUSH {{{1
                                    if
(os.path.exists("%s.signal.dat.flush" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          #print "prueba: ", fichero
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                          # Hago flush del fichero
                                          t.write()
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Compruebo si existe un
fichero de datos
                                          if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                                # Si existe, indico que
no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                                # Si existe comprimido,
indico que no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          else:
                                                # Si no existe, indico
que hay que aÃ±adir la cabecera
                                                cabecera=True
                                          # Si el fichero no estÃ¡
comprimido
                                          if (not reg_comprimido):
                                                # Abre el fichero de
destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                                # Abre el fichero de
origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                          else:
                                                # Abre el fichero de
destino gz

      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
```

```
                                        # Abre el fichero de origen
gz
      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                        # Leo el contenido del origen

      contenido=FILE_ORIG.readlines()
                                        # Transfiero la cabecera
                                        if (cabecera):

      FILE_DEST.write(contenido[0])
                                        # Transfiero todas las lineas
                                        for linea in contenido[1:]:
                                              FILE_DEST.write(linea)
                                        # Cierro el fichero de
destino
                                        FILE_DEST.close()
                                        # Cierro el fichero de origen
                                        FILE_ORIG.close()
                                        # Borro el fichero de origen
                                        if (not reg_comprimido):

      os.unlink("%s.signal.dat" % (fichero))
                                        else:

      os.unlink("%s.signal.dat.gz" % (fichero))
                                        # Borro la solicitud

      os.unlink("%s.signal.dat.flush" % (fichero))
                                        # Indico con un mensaje de
debug el fin de la accion
                                        t.debug("[ADQUIRIR_MODBUS]
FLUSH procesado")
                                  # }}}1
                                  # Compruebo la existencia del
fichero CLEAN {{{1
                                  if
(os.path.exists("%s.signal.dat.clean" % (fichero))):
                                        # Indico con un mensaje de
debug el inicio de la accion
                                        t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                        # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                        # Borro el fichero de datos
                                        if (not reg_comprimido):

      os.unlink("%s.signal.dat.reg" % (fichero))
                                        else:

      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                        # Borro la solicitud

      os.unlink("%s.signal.dat.clean" % (fichero))
                                        # Indico con un mensaje de
debug el fin de la accion
```

```
                                        t.debug("[ADQUIRIR_MODBUS]
CLEAN procesado")
                                    # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 4. Archivo de Configuración adquirir_momentum_M7330_AL1S1.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS V 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                 #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa          #
# VersiÃ³n: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################


#############################################################################
###############################
#################################################(EDITAR)###############
###############################
# Aqui debe declarar el numero de telemando para el cual funcionarÃ
esta libreria adquirir, ya que #
# solo para ese telemando adquirira datos, no olvide que se cuenta en
orden de declaraciÃ²n el en   #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################
###############################
NUM_TELEMANDO=1

#############################################################################
###############################
#############################################################################
###############################
#############################################################################
###############################
#############################################################################
###############################

# Librerias {{{1
# Cargo la configuracion por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# DecisiÃ³n segÃºn parÃ¡metros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
```

```
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=unidades_telemando[NUM_TELEMANDO]

      # listado de divisores del telemando actual
divisores_telemando_actual=divisor_telemando[NUM_TELEMANDO]

# Inicializo los datos
vez=1
end=False
escribir=False

# lista de ids_unidades y relacion de ids con registros
ids_telemando=[]
ids_registros=[]
band=False

try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecucciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # control para cargar el dato en la db.
                  guardar=sql.condicion_tiempo()
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
#                       print "entro 1 guardar=",guardar, "band ",
band," ids_registros=",ids_registros, " divisores
telemando=",divisores_telemando_actual
```

```
#                       time.sleep(1)

      lista_retorno=t.write(guardar,band,ids_registros,divisores_telem
ando_actual)
#                       print "lista de retorno....",lista_retorno
#                       time.sleep(5)
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
#                             print "Error en la AdquisiciÃ²n.
Comprueve la conexiÃ²n y vuelva a reiniciar la adquisiciÃ²n."
#                             time.sleep(2)
                        # Obtengo los ids de la seÃ±al
                              # lista en donde obtengo todos los ids.

                        if (band == False):
                              for datos in lista_retorno:
#                                   print "datos...",datos

      ids=sql.ids_unidades(telemando_actual,datos)
                                    ids_telemando.append(ids[0])
#                                   print "lista de ids=",ids_telemando
#                                   time.sleep(5)

      ids_registros.append((ids[0],ids[1]))
#                                   print "lista de registros
ids=",ids_registros
                                    band=True
#                       print "ids telemando=",ids_telemando
#                       time.sleep(2)

                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,diviso
res_telemando_actual,ids_telemando)
#                             print "unidades del
telemando=",unidades_telemando_actual
#                             print "datos actuales: ", datos
#                             time.sleep(2)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores

#                 print "guardar==",guardar
#                 time.sleep(2)
```

```
                  if (guardar == True):
#                       print "paso......."
#                       time.sleep(2)
                        for registrador in listado_registradores:
                              # Otengo el nombre del fichero de salida
                              try:
                                    # Fichero de salida
                                    fichero=registrador.fichero()
                              except Exception:
                                    # No hay fichero de salida en este
registrador
                                    fichero=None
                              # Si existe fichero de salida, lo proceso
                              if (fichero):
                                    # Compruebo la existencia del
fichero FLUSH {{{1
                                    if
(os.path.exists("%s.signal.dat.flush" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          print "prueba: ", fichero
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                          # Hago flush del fichero
                                          t.write()
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Compruebo si existe un
fichero de datos
                                          if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                                # Si existe, indico que
no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                                # Si existe comprimido,
indico que no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          else:
                                                # Si no existe, indico
que hay que aÃ±adir la cabecera
                                                cabecera=True
                                          # Si el fichero no estÃ¡
comprimido
                                          if (not reg_comprimido):
                                                # Abre el fichero de
destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                                # Abre el fichero de
origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                          else:
                                                # Abre el fichero de
destino gz
```

```
      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
                                              # Abre el fichero de
origen gz

      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                          # Leo el contenido del origen

      contenido=FILE_ORIG.readlines()
                                          # Transfiero la cabecera
                                          if (cabecera):

      FILE_DEST.write(contenido[0])
                                          # Transfiero todas las lineas
                                          for linea in contenido[1:]:
                                                FILE_DEST.write(linea)
                                          # Cierro el fichero de
destino
                                          FILE_DEST.close()
                                          # Cierro el fichero de origen
                                          FILE_ORIG.close()
                                          # Borro el fichero de origen
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.flush" % (fichero))
                                          # Indico con un mensaje de
debug el fin de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
FLUSH procesado")
                                    # }}}1
                                    # Compruebo la existencia del
fichero CLEAN {{{1
                                    if
(os.path.exists("%s.signal.dat.clean" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Borro el fichero de datos
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat.reg" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.clean" % (fichero))
```

```
                                        # Indico con un mensaje de
debug el fin de la accion
                                        t.debug("[ADQUIRIR_MODBUS]
CLEAN procesado")
                                  # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 5. Archivo de Configuración adquirir_momentum_M7330_AL2S1.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS V 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                 #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa          #
# VersiÃ³n: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

#############################################################################
##############################
#################################################(EDITAR)###############
##############################
# Aqui debe declarar el numero de telemando para el cual funcionarÃ
esta libreria adquirir, ya que #
# solo para ese telemando adquirira datos, no olvide que se cuenta en
orden de declaraciÃ²n el en   #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################
##############################
NUM_TELEMANDO=2

#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################

# Librerias {{{1
# Cargo la configuracion por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# DecisiÃ³n segÃºn parÃ¡metros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
```

```
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=unidades_telemando[NUM_TELEMANDO]

      # listado de divisores del telemando actual
divisores_telemando_actual=divisor_telemando[NUM_TELEMANDO]

# Inicializo los datos
vez=1
end=False
escribir=False

# lista de ids_unidades y relacion de ids con registros
ids_telemando=[]
ids_registros=[]
band=False

try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecucciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # control para cargar el dato en la db.
                  guardar=sql.condicion_tiempo()
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
#                       print "entro 1 guardar=",guardar, "band ",
band," ids_registros=",ids_registros, " divisores
telemando=",divisores_telemando_actual
#                       time.sleep(1)
```

```
      lista_retorno=t.write(guardar,band,ids_registros,divisores_telem
ando_actual)
#                       print "lista de retorno....",lista_retorno
#                       time.sleep(5)
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
#                             print "Error en la AdquisiciÃ²n.
Comprueve la conexiÃ²n y vuelva a reiniciar la adquisiciÃ²n."
#                             time.sleep(2)
                        # Obtengo los ids de la seÃ±al
                              # lista en donde obtengo todos los ids.

                        if (band == False):
                              for datos in lista_retorno:
#                                   print "datos...",datos

      ids=sql.ids_unidades(telemando_actual,datos)
                                    ids_telemando.append(ids[0])
#                                   print "lista de ids=",ids_telemando
#                                   time.sleep(5)

      ids_registros.append((ids[0],ids[1]))
#                                   print "lista de registros
ids=",ids_registros
                                    band=True
#                       print "ids telemando=",ids_telemando
#                       time.sleep(2)

                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,diviso
res_telemando_actual,ids_telemando)
#                             print "unidades del
telemando=",unidades_telemando_actual
#                             print "datos actuales: ", datos
#                             time.sleep(2)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores

#                 print "guardar==",guardar
#                 time.sleep(2)
                  if (guardar == True):
```

```
#                       print "paso......."
#                       time.sleep(2)
                        for registrador in listado_registradores:
                              # Otengo el nombre del fichero de salida
                              try:
                                    # Fichero de salida
                                    fichero=registrador.fichero()
                              except Exception:
                                    # No hay fichero de salida en este
registrador
                                    fichero=None
                              # Si existe fichero de salida, lo proceso
                              if (fichero):
                                    # Compruebo la existencia del
fichero FLUSH {{{1
                                    if
(os.path.exists("%s.signal.dat.flush" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          print "prueba: ", fichero
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                          # Hago flush del fichero
                                          t.write()
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Compruebo si existe un
fichero de datos
                                          if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                                # Si existe, indico que
no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                                # Si existe comprimido,
indico que no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          else:
                                                # Si no existe, indico
que hay que aÃ±adir la cabecera
                                                cabecera=True
                                          # Si el fichero no estÃ¡
comprimido
                                          if (not reg_comprimido):
                                                # Abre el fichero de
destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                                # Abre el fichero de
origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                          else:
                                                # Abre el fichero de
destino gz
      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
```

```
                                                # Abre el fichero de
origen gz
      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                          # Leo el contenido del origen
      contenido=FILE_ORIG.readlines()
                                          # Transfiero la cabecera
                                          if (cabecera):
      FILE_DEST.write(contenido[0])
                                          # Transfiero todas las lineas
                                          for linea in contenido[1:]:
                                                FILE_DEST.write(linea)
                                          # Cierro el fichero de
destino
                                          FILE_DEST.close()
                                          # Cierro el fichero de origen
                                          FILE_ORIG.close()
                                          # Borro el fichero de origen
                                          if (not reg_comprimido):
      os.unlink("%s.signal.dat" % (fichero))
                                          else:
      os.unlink("%s.signal.dat.gz" % (fichero))
                                          # Borro la solicitud
      os.unlink("%s.signal.dat.flush" % (fichero))
                                          # Indico con un mensaje de
debug el fin de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
FLUSH procesado")
                                    # }}}1
                                    # Compruebo la existencia del
fichero CLEAN {{{1
                                    if
(os.path.exists("%s.signal.dat.clean" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                          # Compruebo si los datos
estan comprimidos
      reg_comprimido=registrador.compress()
                                          # Borro el fichero de datos
                                          if (not reg_comprimido):
      os.unlink("%s.signal.dat.reg" % (fichero))
                                          else:
      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                          # Borro la solicitud
      os.unlink("%s.signal.dat.clean" % (fichero))
                                          # Indico con un mensaje de
debug el fin de la accion
```

```
                                        t.debug("[ADQUIRIR_MODBUS]
CLEAN procesado")
                                  # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 6. Archivo de Configuración adquirir_momentum_M7330_AL3S1.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS V 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                 #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa          #
# VersiÃ³n: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

#############################################################################
##############################
#################################################(EDITAR)###############
##############################
# Aqui debe declarar el numero de telemando para el cual funcionarÃ
esta libreria adquirir, ya que #
# solo para ese telemando adquirira datos, no olvide que se cuenta en
orden de declaraciÃ²n el en   #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################
##############################
NUM_TELEMANDO=3

#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################

# Librerias {{{1
# Cargo la configuracion por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# DecisiÃ³n segÃºn parÃ¡metros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
```

```
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=unidades_telemando[NUM_TELEMANDO]

      # listado de divisores del telemando actual
divisores_telemando_actual=divisor_telemando[NUM_TELEMANDO]

# Inicializo los datos
vez=1
end=False
escribir=False

# lista de ids_unidades y relacion de ids con registros
ids_telemando=[]
ids_registros=[]
band=False

try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecucciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # control para cargar el dato en la db.
                  guardar=sql.condicion_tiempo()
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
#                       print "entro 1 guardar=",guardar, "band ",
band," ids_registros=",ids_registros, " divisores
telemando=",divisores_telemando_actual
#                       time.sleep(1)
```

```
      lista_retorno=t.write(guardar,band,ids_registros,divisores_telem
ando_actual)
#                       print "lista de retorno....",lista_retorno
#                       time.sleep(5)
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
#                             print "Error en la AdquisiciÃ²n.
Comprueve la conexiÃ²n y vuelva a reiniciar la adquisiciÃ²n."
#                             time.sleep(2)
                        # Obtengo los ids de la seÃ±al
                              # lista en donde obtengo todos los ids.

                        if (band == False):
                              for datos in lista_retorno:
#                                   print "datos...",datos

      ids=sql.ids_unidades(telemando_actual,datos)
                                    ids_telemando.append(ids[0])
#                                   print "lista de ids=",ids_telemando
#                                   time.sleep(5)

      ids_registros.append((ids[0],ids[1]))
#                                   print "lista de registros
ids=",ids_registros
                                    band=True
#                       print "ids telemando=",ids_telemando
#                       time.sleep(2)

                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,diviso
res_telemando_actual,ids_telemando)
#                             print "unidades del
telemando=",unidades_telemando_actual
#                             print "datos actuales: ", datos
#                             time.sleep(2)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores

#                 print "guardar==",guardar
#                 time.sleep(2)
                  if (guardar == True):
```

```
#                       print "paso......."
#                       time.sleep(2)
                        for registrador in listado_registradores:
                              # Otengo el nombre del fichero de salida
                              try:
                                    # Fichero de salida
                                    fichero=registrador.fichero()
                              except Exception:
                                    # No hay fichero de salida en este
registrador
                                    fichero=None
                              # Si existe fichero de salida, lo proceso
                              if (fichero):
                                    # Compruebo la existencia del
fichero FLUSH {{{1
                                    if
(os.path.exists("%s.signal.dat.flush" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          print "prueba: ", fichero
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                          # Hago flush del fichero
                                          t.write()
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Compruebo si existe un
fichero de datos
                                          if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                                # Si existe, indico que
no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                                # Si existe comprimido,
indico que no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          else:
                                                # Si no existe, indico
que hay que aÃ±adir la cabecera
                                                cabecera=True
                                          # Si el fichero no estÃ¡
comprimido
                                          if (not reg_comprimido):
                                                # Abre el fichero de
destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                                # Abre el fichero de
origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                          else:
                                                # Abre el fichero de
destino gz
```

```
      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
                                              # Abre el fichero de
origen gz

      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                          # Leo el contenido del origen

      contenido=FILE_ORIG.readlines()
                                          # Transfiero la cabecera
                                          if (cabecera):

      FILE_DEST.write(contenido[0])
                                          # Transfiero todas las lineas
                                          for linea in contenido[1:]:
                                                FILE_DEST.write(linea)
                                          # Cierro el fichero de
destino
                                          FILE_DEST.close()
                                          # Cierro el fichero de origen
                                          FILE_ORIG.close()
                                          # Borro el fichero de origen
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.flush" % (fichero))
                                          # Indico con un mensaje de
debug el fin de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
FLUSH procesado")
                                    # }}}1
                                    # Compruebo la existencia del
fichero CLEAN {{{1
                                    if
(os.path.exists("%s.signal.dat.clean" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Borro el fichero de datos
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat.reg" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.clean" % (fichero))
```

```
                                        # Indico con un mensaje de
debug el fin de la accion
                                        t.debug("[ADQUIRIR_MODBUS]
CLEAN procesado")
                                  # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 7. Archivo de Configuración adquirir_momentum_M7330_AL4S1.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS V 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                 #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa          #
# VersiÃ³n: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

#############################################################################
##############################
#################################################(EDITAR)###############
##############################
# Aqui debe declarar el numero de telemando para el cual funcionarÃ
esta libreria adquirir, ya que #
# solo para ese telemando adquirira datos, no olvide que se cuenta en
orden de declaraciÃ²n el en   #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################
##############################
NUM_TELEMANDO=4

#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################

# Librerias {{{1
# Cargo la configuracion por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# DecisiÃ³n segÃºn parÃ¡metros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
```

```
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=unidades_telemando[NUM_TELEMANDO]

      # listado de divisores del telemando actual
divisores_telemando_actual=divisor_telemando[NUM_TELEMANDO]

# Inicializo los datos
vez=1
end=False
escribir=False

# lista de ids_unidades y relacion de ids con registros
ids_telemando=[]
ids_registros=[]
band=False

try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecucciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # control para cargar el dato en la db.
                  guardar=sql.condicion_tiempo()
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
#                       print "entro 1 guardar=",guardar, "band ",
band," ids_registros=",ids_registros, " divisores
telemando=",divisores_telemando_actual
#                       time.sleep(1)
```

```
      lista_retorno=t.write(guardar,band,ids_registros,divisores_telem
ando_actual)
#                       print "lista de retorno....",lista_retorno
#                       time.sleep(5)
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
#                             print "Error en la AdquisiciÃ²n.
Comprueve la conexiÃ²n y vuelva a reiniciar la adquisiciÃ²n."
#                             time.sleep(2)
                        # Obtengo los ids de la seÃ±al
                              # lista en donde obtengo todos los ids.

                        if (band == False):
                              for datos in lista_retorno:
#                                   print "datos...",datos

      ids=sql.ids_unidades(telemando_actual,datos)
                                    ids_telemando.append(ids[0])
#                                   print "lista de ids=",ids_telemando
#                                   time.sleep(5)

      ids_registros.append((ids[0],ids[1]))
#                                   print "lista de registros
ids=",ids_registros
                                    band=True
#                       print "ids telemando=",ids_telemando
#                       time.sleep(2)

                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,diviso
res_telemando_actual,ids_telemando)
#                             print "unidades del
telemando=",unidades_telemando_actual
#                             print "datos actuales: ", datos
#                             time.sleep(2)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores

#                 print "guardar==",guardar
#                 time.sleep(2)
                  if (guardar == True):
```

```
#                       print "paso......."
#                       time.sleep(2)
                        for registrador in listado_registradores:
                              # Otengo el nombre del fichero de salida
                              try:
                                    # Fichero de salida
                                    fichero=registrador.fichero()
                              except Exception:
                                    # No hay fichero de salida en este
registrador
                                    fichero=None
                              # Si existe fichero de salida, lo proceso
                              if (fichero):
                                    # Compruebo la existencia del
fichero FLUSH {{{1
                                    if
(os.path.exists("%s.signal.dat.flush" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          print "prueba: ", fichero
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                          # Hago flush del fichero
                                          t.write()
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Compruebo si existe un
fichero de datos
                                          if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                                # Si existe, indico que
no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                                # Si existe comprimido,
indico que no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          else:
                                                # Si no existe, indico
que hay que aÃ±adir la cabecera
                                                cabecera=True
                                          # Si el fichero no estÃ¡
comprimido
                                          if (not reg_comprimido):
                                                # Abre el fichero de
destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                                # Abre el fichero de
origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                          else:
                                                # Abre el fichero de
destino gz
```

```
      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
                                              # Abre el fichero de
origen gz

      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                          # Leo el contenido del origen

      contenido=FILE_ORIG.readlines()
                                          # Transfiero la cabecera
                                          if (cabecera):

      FILE_DEST.write(contenido[0])
                                          # Transfiero todas las lineas
                                          for linea in contenido[1:]:
                                                FILE_DEST.write(linea)
                                          # Cierro el fichero de
destino
                                          FILE_DEST.close()
                                          # Cierro el fichero de origen
                                          FILE_ORIG.close()
                                          # Borro el fichero de origen
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.flush" % (fichero))
                                          # Indico con un mensaje de
debug el fin de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
FLUSH procesado")
                                    # }}}1
                                    # Compruebo la existencia del
fichero CLEAN {{{1
                                    if
(os.path.exists("%s.signal.dat.clean" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Borro el fichero de datos
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat.reg" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.clean" % (fichero))
```

```
                                        # Indico con un mensaje de
debug el fin de la accion
                                        t.debug("[ADQUIRIR_MODBUS]
CLEAN procesado")
                                  # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 8. Archivo de Configuración adquirir_momentum_M7330_AL5S1.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS V 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                 #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa          #
# VersiÃ³n: 0000000001                                      #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

#############################################################################
##############################
#################################################(EDITAR)###############
##############################
# Aqui debe declarar el numero de telemando para el cual funcionarÃ
esta libreria adquirir, ya que #
# solo para ese telemando adquirira datos, no olvide que se cuenta en
orden de declaraciÃ²n el en   #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################
##############################
NUM_TELEMANDO=5

#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################
#############################################################################
##############################

# Librerias {{{1
# Cargo la configuracion por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# DecisiÃ³n segÃºn parÃ¡metros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
```

```
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=unidades_telemando[NUM_TELEMANDO]

      # listado de divisores del telemando actual
divisores_telemando_actual=divisor_telemando[NUM_TELEMANDO]

# Inicializo los datos
vez=1
end=False
escribir=False

# lista de ids_unidades y relacion de ids con registros
ids_telemando=[]
ids_registros=[]
band=False

try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecucciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # control para cargar el dato en la db.
                  guardar=sql.condicion_tiempo()
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
#                       print "entro 1 guardar=",guardar, "band ",
band," ids_registros=",ids_registros, " divisores
telemando=",divisores_telemando_actual
#                       time.sleep(1)
```

```
      lista_retorno=t.write(guardar,band,ids_registros,divisores_telem
ando_actual)
#                       print "lista de retorno....",lista_retorno
#                       time.sleep(5)
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
#                             print "Error en la AdquisiciÃ²n.
Comprueve la conexiÃ²n y vuelva a reiniciar la adquisiciÃ²n."
#                             time.sleep(2)
                        # Obtengo los ids de la seÃ±al
                              # lista en donde obtengo todos los ids.

                        if (band == False):
                              for datos in lista_retorno:
#                                   print "datos...",datos

      ids=sql.ids_unidades(telemando_actual,datos)
                                    ids_telemando.append(ids[0])
#                                   print "lista de ids=",ids_telemando
#                                   time.sleep(5)

      ids_registros.append((ids[0],ids[1]))
#                                   print "lista de registros
ids=",ids_registros
                                    band=True
#                       print "ids telemando=",ids_telemando
#                       time.sleep(2)

                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,diviso
res_telemando_actual,ids_telemando)
#                             print "unidades del
telemando=",unidades_telemando_actual
#                             print "datos actuales: ", datos
#                             time.sleep(2)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores

#                 print "guardar==",guardar
#                 time.sleep(2)
                  if (guardar == True):
```

```
#                       print "paso......."
#                       time.sleep(2)
                        for registrador in listado_registradores:
                              # Otengo el nombre del fichero de salida
                              try:
                                    # Fichero de salida
                                    fichero=registrador.fichero()
                              except Exception:
                                    # No hay fichero de salida en este
registrador
                                    fichero=None
                              # Si existe fichero de salida, lo proceso
                              if (fichero):
                                    # Compruebo la existencia del
fichero FLUSH {{{1
                                    if
(os.path.exists("%s.signal.dat.flush" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          print "prueba: ", fichero
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                          # Hago flush del fichero
                                          t.write()
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Compruebo si existe un
fichero de datos
                                          if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                                # Si existe, indico que
no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                                # Si existe comprimido,
indico que no hay que aÃ±adir la cabecera
                                                cabecera=False
                                          else:
                                                # Si no existe, indico
que hay que aÃ±adir la cabecera
                                                cabecera=True
                                          # Si el fichero no estÃ¡
comprimido
                                          if (not reg_comprimido):
                                                # Abre el fichero de
destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                                # Abre el fichero de
origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                          else:
                                                # Abre el fichero de
destino gz
```

```
      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
                                              # Abre el fichero de
origen gz

      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                          # Leo el contenido del origen

      contenido=FILE_ORIG.readlines()
                                          # Transfiero la cabecera
                                          if (cabecera):

      FILE_DEST.write(contenido[0])
                                          # Transfiero todas las lineas
                                          for linea in contenido[1:]:
                                                FILE_DEST.write(linea)
                                          # Cierro el fichero de
destino
                                          FILE_DEST.close()
                                          # Cierro el fichero de origen
                                          FILE_ORIG.close()
                                          # Borro el fichero de origen
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.flush" % (fichero))
                                          # Indico con un mensaje de
debug el fin de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
FLUSH procesado")
                                    # }}}1
                                    # Compruebo la existencia del
fichero CLEAN {{{1
                                    if
(os.path.exists("%s.signal.dat.clean" % (fichero))):
                                          # Indico con un mensaje de
debug el inicio de la accion
                                          t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                          # Compruebo si los datos
estan comprimidos

      reg_comprimido=registrador.compress()
                                          # Borro el fichero de datos
                                          if (not reg_comprimido):

      os.unlink("%s.signal.dat.reg" % (fichero))
                                          else:

      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                          # Borro la solicitud

      os.unlink("%s.signal.dat.clean" % (fichero))
```

```
                                        # Indico con un mensaje de
debug el fin de la accion
                                        t.debug("[ADQUIRIR_MODBUS]
CLEAN procesado")
                                  # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 9. Archivo de Configuración adquirir_momentum_M7330_AL6S1.py

```
#!/usr/bin/env python2.4
#-*- coding: utf-8 -*-
############################################################
#               ADQUISITOR DE DATOS v1.0                   #
############################################################
# Autor: Francisco Narváez                                 #
#        Wilmer Guamaní                                    #
# Fecha: Szczecin, 05 de Enero del 2011                    #
# Descripción: Fichero de configuración de Adquisición de#
# Datos mediante MODBUS                                    #
# Versión: 2006123001                                      #
#                                                          #
# Código fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com                                #
# Wilmer_g_p@hotmail.com                                   #
############################################################

################################(EDITAR)#####################################
# Aquí debe declarar el numero de telemando para el cual funcionará
esta librería adquirir, ya que #
# Solo para ese telemando adquirirá datos, no olvide que se cuenta en
orden de declaración en el  #
# archivo /etc/adquirir_momentum.py , y la cuenta comienza en cero
#
#############################################################################

NUM_TELEMANDO=6

#############################################################################

# Librerías {{{1
# Cargo la configuración por defecto
from src.etcbin.cargador_sql_EERSA import *
from src.etcbin.adquirir_momentum import *
import os
import sys
# }}}1

# Decisión según parámetros de entrada
if ((len(sys.argv)==2) and (sys.argv[1]=='--silent')):
      debug=False
else:
      debug=True

# Cargo el estado de debug en los Telemandos
#for t in listado_telemandos:
      # id inicial del telemando actual
num_senales_ini=num_senales_telemando_ini[NUM_TELEMANDO]
      # id final del telemando actual
num_senales_fin=num_senales_telemando_fin[NUM_TELEMANDO]
      # Telemando actual
t=listado_telemandos[NUM_TELEMANDO]
t.set_debug(debug)

      # Nombre del telemando actual
```

```
telemando_actual=nombre_telemandos[NUM_TELEMANDO]

      # listado de unidades del telemando actual
unidades_telemando_actual=[]
unidades_telemando_actual.append(unidades_telemando[0])

# Inicializo los datos
vez=1
end=False
escribir=False
try:
      # Ejecutar siempre
      modbus=t.conectar()
      sql.senales(s)
      sql.coneccion_database()
      while (not end):
            # Espero 5 segundos
            time.sleep(dormir)
            # Control de ejecuciones
            if (vez==veces):
                  # Inicializo vez
                  vez=1
                  # Escribe a disco ahora
                  escribir=True
            else:
                  # Incremento una vez
                  vez=vez+1
                  # No escribas a disco ahora
                  escribir=False
            # Para todos los Telemandos
#           for t in listado_telemandos1:
            try:
                  # Descargo datos del Telemando
                  t.execute(modbus)
                  # Control de escritura a disco
                  if (escribir):
                        # Escribo a fichero y tengo  la lista de datos
adquiridos
                        lista_retorno=t.write()
#                       print "lista de datos retornados=
",lista_retorno
#                       time.sleep(5)
                        #Cargando los datos en la base de Datos
                        sql.crear_tabla(telemando_actual)
#                       print "lista de retorno=",lista_retorno
                        sql.borrar_tabla(telemando_actual)
#,num_senales_ini,num_senales_fin
                        if (len(lista_retorno) == 0):

      sql.load_error(telemando_actual,num_senales_ini,num_senales_fin)
                              print "Error en la Adquisiciòn. Compruebe
la conexión y vuelva a reiniciar la adquisición."
                              time.sleep(2)
                        # Obtengo los ids de la señal
                              # lista en donde obtengo todos los ids.
                        ids_telemando=[]
                        for datos in lista_retorno:

      ids=sql.ids_unidades(telemando_actual,datos)
```

```
                              ids_telemando.append(ids)
                              print "lista de ids=",ids_telemando
                        for datos in lista_retorno:
#                             print "Datos insertados=",datos

      sql.load(telemando_actual,datos,unidades_telemando_actual,ids_te
lemando)
#                             time.sleep(dormir)
#                       print "Datos insertados....."
#                       time.sleep(10)
#                       print "ESPERA....................."
#                       time.sleep(10)
                        t.vaciar_retorno()
                  # Obtengo el listado de registradores
                  listado_registradores=t.registradores()
                  # Para el listado de registradores
                  for registrador in listado_registradores:
                        # Obtengo el nombre del fichero de salida
                        try:
                              # Fichero de salida
                              fichero=registrador.fichero()
                        except Exception:
                              # No hay fichero de salida en este
registrador
                              fichero=None
                        # Si existe fichero de salida, lo proceso
                        if (fichero):
                              # Compruebo la existencia del fichero
FLUSH {{{1
                              if (os.path.exists("%s.signal.dat.flush"
% (fichero))):
                                    # Indico con un mensaje de debug el
inicio de la acción
                                    print "prueba: ", fichero
                                    t.debug("[ADQUIRIR_MODBUS]
Procesando FLUSH")
                                    # Hago flush del fichero
                                    t.write()
                                    # Compruebo si los datos están
comprimidos

      reg_comprimido=registrador.compress()
                                    # Compruebo si existe un fichero de
datos
                                    if ((not reg_comprimido) and
(os.path.exists("%s.signal.dat.reg" % (fichero)))):
                                          # Si existe, indico que no
hay que añadir la cabecera
                                          cabecera=False
                                    elif ((reg_comprimido) and
(os.path.exists("%s.signal.dat.reg.gz" % (fichero)))):
                                          # Si existe comprimido,
indico que no hay que añadir la cabecera
                                          cabecera=False
                                    else:
                                          # Si no existe, indico que
hay que añadir la cabecera
                                          cabecera=True
                                    # Si el fichero no está comprimido
```

```
                                    if (not reg_comprimido):
                                          # Abre el fichero de destino

      FILE_DEST=open("%s.signal.dat.reg" % (fichero),'a')
                                          # Abre el fichero de origen

      FILE_ORIG=open("%s.signal.dat" % (fichero),'r')
                                    else:
                                          # Abre el fichero de destino
gz

      FILE_DEST=open("%s.signal.dat.reg.gz" % (fichero),'a')
                                          # Abre el fichero de origen
gz

      FILE_ORIG=open("%s.signal.dat.gz" % (fichero),'r')
                                    # Leo el contenido del origen
                                    contenido=FILE_ORIG.readlines()
                                    # Transfiero la cabecera
                                    if (cabecera):
                                          FILE_DEST.write(contenido[0])
                                    # Transfiero todas las líneas
                                    for linea in contenido[1:]:
                                          FILE_DEST.write(linea)
                                    # Cierro el fichero de destino
                                    FILE_DEST.close()
                                    # Cierro el fichero de origen
                                    FILE_ORIG.close()
                                    # Borro el fichero de origen
                                    if (not reg_comprimido):
                                          os.unlink("%s.signal.dat" %
(fichero))
                                    else:
                                          os.unlink("%s.signal.dat.gz"
% (fichero))
                                    # Borro la solicitud
                                    os.unlink("%s.signal.dat.flush" %
(fichero))
                                    # Indico con un mensaje de debug el
fin de la acción
                                    t.debug("[ADQUIRIR_MODBUS] FLUSH
procesado")
                              # }}}1
                              # Compruebo la existencia del fichero
CLEAN {{{1
                              if (os.path.exists("%s.signal.dat.clean"
% (fichero))):
                                    # Indico con un mensaje de debug el
inicio de la acción
                                    t.debug("[ADQUIRIR_MODBUS]
Procesando CLEAN")
                                    # Compruebo si los datos están
comprimidos

      reg_comprimido=registrador.compress()
                                    # Borro el fichero de datos
                                    if (not reg_comprimido):
                                          os.unlink("%s.signal.dat.reg"
% (fichero))
```

```
                                    else:

      os.unlink("%s.signal.dat.reg.gz" % (fichero))
                                    # Borro la solicitud
                                    os.unlink("%s.signal.dat.clean" %
(fichero))
                                    # Indico con un mensaje de debug el
fin de la acción
                                    t.debug("[ADQUIRIR_MODBUS] CLEAN
procesado")
                              # }}}1
            except KeyboardInterrupt:
                  raise
            except Exception,e:
                  # Muestro el error
                  print "Error descargando del Telemando %s : %s" %
(t.internal_id(),e)
                  # Marco la salida
                  end=True
                  # Termino el bucle
                  break
except KeyboardInterrupt,e:
      # Muestro el mensaje de salida
      print "Saliendo del programa a peticion del usuario..."
      # Escribo los resultados
#     for t in listado_telemandos1:
      t.write()
      sql.desconectar_database()
      t.desconectar(modbus)
      # Muestro mensaje de fin
      print "Escritura finalizada"
```

Anexo 10. Archivo de configuración /etc/señales.ods

Descripción	A/D	Frecuencia	En BD	RTU	ID	Dirección
Ion7650_T1_Vln a	a	60h	SI	M7650T1	M7650T1_Vlna	40167
ion7650_T1_Vln b	a	60h	SI	M7650T1	M7650T1_Vlnb	40169
ion7650_T1_Vln c	a	60h	SI	M7650T1	M7650T1_Vlnc	40171
Ion7650_T1_Vln avg	a	60h	SI	M7650T1	M7650T1_Vlnavg	40173
Ion7650_T1_Vll ab	a	60h	SI	M7650T1	M7650T1_Vllab	40177
ion7650_T1_Vll bc	a	60h	SI	M7650T1	M7650T1_Vllbc	40179
ion7650_T1_Vll ca	a	60h	SI	M7650T1	M7650T1_Vllca	40181
ion7650_T1_Vll avg	a	60h	SI	M7650T1	M7650T1_Vllavg	40183
ion7650_T1_I a	a	60h	SI	M7650T1	M7650T1_Ia	40150
ion7650_T1_I b	a	60h	SI	M7650T1	M7650T1_Ib	40151
ion7650_T1_I c	a	60h	SI	M7650T1	M7650T1_Ic	40152
ion7650_T1_I avg	a	60h	SI	M7650T1	M7650T1_Iavg	40155
ion7650_T1_Freq	a	60h	SI	M7650T1	M7650T1_Freq	40159
ion7650_T1_KW a (Pa)	a	60h	SI	M7650T1	M7650T1_KWa	40199
ion7650_T1_KW b (Pb)	a	60h	SI	M7650T1	M7650T1_KWb	40201
ion7650_T1_KW c (Pc)	a	60h	SI	M7650T1	M7650T1_KWc	40203
ion7650_T1_KW tot (Ptot)	a	60h	SI	M7650T1	M7650T1_KWtot	40205
ion7650_T1_KVAR tot (Qtot)	a	60h	SI	M7650T1	M7650T1_KVARtot	40215
ion7650_T1_KVA tot (Stot)	a	60h	SI	M7650T1	M7650T1_KVAtot	40225
ion7650_T1_PFsign tot	a	60h	SI	M7650T1	M7650T1_PFsigntot	40265
ion7330_AL1_S1_Vln a	a	60h	SI	M7330AL1S1	M7330AL1S1_Vlna	40011
ion7330_AL1_S1_Vln b	a	60h	SI	M7330AL1S1	M7330AL1S1_Vlnb	40012
ion7330_AL1_S1_Vln c	a	60h	SI	M7330AL1S1	M7330AL1S1_Vlnc	40013
ion7330_AL1_S1_Vln avg	a	60h	SI	M7330AL1S1	M7330AL1S1_Vlnavg	40014
ion7330_AL1_S1_Vll ab	a	60h	SI	M7330AL1S1	M7330AL1S1_Vllab	40015

ion7330_AL1_S1_Vll bc	a	60h	SI	M7330AL1S1	M7330AL1S1_Vllbc	40016
ion7330_AL1_S1_Vll ca	a	60h	SI	M7330AL1S1	M7330AL1S1_Vllca	40017
ion7330_AL1_S1_Vll avg	a	60h	SI	M7330AL1S1	M7330AL1S1_Vllavg	40018
ion7330_AL1_S1_I a	a	60h	SI	M7330AL1S1	M7330AL1S1_Ia	40019
ion7330_AL1_S1_I b	a	60h	SI	M7330AL1S1	M7330AL1S1_Ib	40020
ion7330_AL1_S1_I c	a	60h	SI	M7330AL1S1	M7330AL1S1_Ic	40021
ion7330_AL1_S1_I avg	a	60h	SI	M7330AL1S1	M7330AL1S1_Iavg	40022
ion7330_AL1_S1_Freq	a	60h	SI	M7330AL1S1	M7330AL1S1_Freq	40025
ion7330_AL1_S1_KW a (Pa)	a	60h	SI	M7330AL1S1	M7330AL1S1_KWa	40028
ion7330_AL1_S1_KW b (Pb)	a	60h	SI	M7330AL1S1	M7330AL1S1_KWb	40030
ion7330_AL1_S1_KW c (Pc)	a	60h	SI	M7330AL1S1	M7330AL1S1_KWc	40032
ion7330_AL1_S1_KW tot (Ptot)	a	60h	SI	M7330AL1S1	M7330AL1S1_KWtot	40034
ion7330_AL1_S1_KVAR tot (Qtot)	a	60h	SI	M7330AL1S1	M7330AL1S1_KVARtot	40042
ion7330_AL1_S1_KVA tot (Stot)	a	60h	SI	M7330AL1S1	M7330AL1S1_KVAtot	40050
ion7333_AL1_S1_PFsign tot	a	60h	SI	M7330AL1S1	M7330AL1S1_PFsigntot	40058
ion7330_AL2_S1_Vln a	a	60h	SI	M7330AL2S1	M7330AL2S1_Vlna	40011
ion7330_AL2_S1_Vln b	a	60h	SI	M7330AL2S1	M7330AL2S1_Vlnb	40012
ion7330_AL2_S1_Vln c	a	60h	SI	M7330AL2S1	M7330AL2S1_Vlnc	40013
ion7330_AL2_S1_Vln avg	a	60h	SI	M7330AL2S1	M7330AL2S1_Vlnavg	40014
ion7330_AL2_S1_Vll ab	a	60h	SI	M7330AL2S1	M7330AL2S1_Vllab	40015
ion7330_AL2_S1_Vll bc	a	60h	SI	M7330AL2S1	M7330AL2S1_Vllbc	40016
ion7330_AL2_S1_Vll ca	a	60h	SI	M7330AL2S1	M7330AL2S1_Vllca	40017
ion7330_AL2_S1_Vll avg	a	60h	SI	M7330AL2S1	M7330AL2S1_Vllavg	40018
ion7330_AL2_S1_I a	a	60h	SI	M7330AL2S1	M7330AL2S1_Ia	40019
ion7330_AL2_S1_I b	a	60h	SI	M7330AL2S1	M7330AL2S1_Ib	40020
ion7330_AL2_S1_I c	a	60h	SI	M7330AL2S1	M7330AL2S1_Ic	40021
ion7330_AL2_S1_I avg	a	60h	SI	M7330AL2S1	M7330AL2S1_Iavg	40022
ion7330_AL2_S1_Freq	a	60h	SI	M7330AL2S1	M7330AL2S1_Freq	40025
ion7330_AL2_S1_KW a (Pa)	a	60h	SI	M7330AL2S1	M7330AL2S1_KWa	40028
ion7330_AL2_S1_KW b (Pb)	a	60h	SI	M7330AL2S1	M7330AL2S1_KWb	40030

ion7330_AL2_S1_KW c (Pc)	a	60h	SI	M7330AL2S1	M7330AL2S1_KWc	40032
ion7330_AL2_S1_KW tot (Ptot)	a	60h	SI	M7330AL2S1	M7330AL2S1_KWtot	40034
ion7330_AL2_S1_KVAR tot (Qtot)	a	60h	SI	M7330AL2S1	M7330AL2S1_KVARtot	40042
ion7330_AL2_S1_KVA tot (Stot)	a	60h	SI	M7330AL2S1	M7330AL2S1_KVAtot	40050
ion7330_AL2_S1_PFsign tot	a	60h	SI	M7330AL2S1	M7330AL2S1_PFsigntot	40058
ion7330_AL3_S1_Vln a	a	60h	SI	M7330AL3S1	M7330AL3S1_Vlna	40011
ion7330_AL3_S1_Vln b	a	60h	SI	M7330AL3S1	M7330AL3S1_Vlnb	40012
ion7330_AL3_S1_Vln c	a	60h	SI	M7330AL3S1	M7330AL3S1_Vlnc	40013
ion7330_AL3_S1_Vln avg	a	60h	SI	M7330AL3S1	M7330AL3S1_Vlnavg	40014
ion7330_AL3_S1_Vll ab	a	60h	SI	M7330AL3S1	M7330AL3S1_Vllab	40015
ion7330_AL3_S1_Vll bc	a	60h	SI	M7330AL3S1	M7330AL3S1_Vllbc	40016
ion7330_AL3_S1_Vll ca	a	60h	SI	M7330AL3S1	M7330AL3S1_Vllca	40017
ion7330_AL3_S1_Vll avg	a	60h	SI	M7330AL3S1	M7330AL3S1_Vllavg	40018
ion7330_AL3_S1_I a	a	60h	SI	M7330AL3S1	M7330AL3S1_Ia	40019
ion7330_AL3_S1_I b	a	60h	SI	M7330AL3S1	M7330AL3S1_Ib	40020
ion7330_AL3_S1_I c	a	60h	SI	M7330AL3S1	M7330AL3S1_Ic	40021
ion7330_AL3_S1_I avg	a	60h	SI	M7330AL3S1	M7330AL3S1_Iavg	40022
ion7330_AL3_S1_Freq	a	60h	SI	M7330AL3S1	M7330AL3S1_Freq	40025
ion7330_AL3_S1_KW a (Pa)	a	60h	SI	M7330AL3S1	M7330AL3S1_KWa	40028
ion7330_AL3_S1_KW b (Pb)	a	60h	SI	M7330AL3S1	M7330AL3S1_KWb	40030
ion7330_AL3_S1_KW c (Pc)	a	60h	SI	M7330AL3S1	M7330AL3S1_KWc	40032
ion7330_AL3_S1_KW tot (Ptot)	a	60h	SI	M7330AL3S1	M7330AL3S1_KWtot	40034
ion7330_AL3_S1_KVAR tot (Qtot)	a	60h	SI	M7330AL3S1	M7330AL3S1_KVARtot	40042
ion7330_AL3_S1_KVA tot (Stot)	a	60h	SI	M7330AL3S1	M7330AL3S1_KVAtot	40050
ion7330_AL3_S1_PFsign tot	a	60h	SI	M7330AL3S1	M7330AL3S1_PFsigntot	40058
ion7330_AL4_S1_Vln a	a	60h	SI	M7330AL4S1	M7330AL4S1_Vlna	40011
ion7330_AL4_S1_Vln b	a	60h	SI	M7330AL4S1	M7330AL4S1_Vlnb	40012
ion7330_AL4_S1_Vln c	a	60h	SI	M7330AL4S1	M7330AL4S1_Vlnc	40013

ion7330_AL4_S1_Vln avg	a	60h	SI	M7330AL4S1	M7330AL4S1_Vlnavg	40014
ion7330_AL4_S1_Vll ab	a	60h	SI	M7330AL4S1	M7330AL4S1_Vllab	40015
ion7330_AL4_S1_Vll bc	a	60h	SI	M7330AL4S1	M7330AL4S1_Vllbc	40016
ion7330_AL4_S1_Vll ca	a	60h	SI	M7330AL4S1	M7330AL4S1_Vllca	40017
ion7330_AL4_S1_Vll avg	a	60h	SI	M7330AL4S1	M7330AL4S1_Vllavg	40018
ion7330_AL4_S1_I a	a	60h	SI	M7330AL4S1	M7330AL4S1_Ia	40019
ion7330_AL4_S1_I b	a	60h	SI	M7330AL4S1	M7330AL4S1_Ib	40020
ion7330_AL4_S1_I c	a	60h	SI	M7330AL4S1	M7330AL4S1_Ic	40021
ion7330_AL4_S1_I avg	a	60h	SI	M7330AL4S1	M7330AL4S1_Iavg	40022
ion7330_AL4_S1_Freq	a	60h	SI	M7330AL4S1	M7330AL4S1_Freq	40025
ion7330_AL4_S1_KW a (Pa)	a	60h	SI	M7330AL4S1	M7330AL4S1_KWa	40028
ion7330_AL4_S1_KW b (Pb)	a	60h	SI	M7330AL4S1	M7330AL4S1_KWb	40030
ion7330_AL4_S1_KW c (Pc)	a	60h	SI	M7330AL4S1	M7330AL4S1_KWc	40032
ion7330_AL4_S1_KW tot (Ptot)	a	60h	SI	M7330AL4S1	M7330AL4S1_KWtot	40034
ion7330_AL4_S1_KVAR tot (Qtot)	a	60h	SI	M7330AL4S1	M7330AL4S1_KVARtot	40042
ion7330_AL4_S1_KVA tot (Stot)	a	60h	SI	M7330AL4S1	M7330AL4S1_KVAtot	40050
ion7330_AL4_S1_PFsign tot	a	60h	SI	M7330AL4S1	M7330AL4S1_PFsigntot	40058
ion7330_AL5_S1_Vln a	a	60h	SI	M7330AL5S1	M7330AL5S1_Vlna	40011
ion7330_AL5_S1_Vln b	a	60h	SI	M7330AL5S1	M7330AL5S1_Vlnb	40012
ion7330_AL5_S1_Vln c	a	60h	SI	M7330AL5S1	M7330AL5S1_Vlnc	40013
ion7330_AL5_S1_Vln avg	a	60h	SI	M7330AL5S1	M7330AL5S1_Vlnavg	40014
ion7330_AL5_S1_Vll ab	a	60h	SI	M7330AL5S1	M7330AL5S1_Vllab	40015
ion7330_AL5_S1_Vll bc	a	60h	SI	M7330AL5S1	M7330AL5S1_Vllbc	40016
ion7330_AL5_S1_Vll ca	a	60h	SI	M7330AL5S1	M7330AL5S1_Vllca	40017
ion7330_AL5_S1_Vll avg	a	60h	SI	M7330AL5S1	M7330AL5S1_Vllavg	40018
ion7330_AL5_S1_I a	a	60h	SI	M7330AL5S1	M7330AL5S1_Ia	40019
ion7330_AL5_S1_I b	a	60h	SI	M7330AL5S1	M7330AL5S1_Ib	40020
ion7330_AL5_S1_I c	a	60h	SI	M7330AL5S1	M7330AL5S1_Ic	40021
ion7330_AL5_S1_I avg	a	60h	SI	M7330AL5S1	M7330AL5S1_Iavg	40022
ion7330_AL5_S1_Freq	a	60h	SI	M7330AL5S1	M7330AL5S1_Freq	40025

ion7330_AL5_S1_KW a (Pa)	a	60h	SI	M7330AL5S1	M7330AL5S1_KWa	40028
ion7330_AL5_S1_KW b (Pb)	a	60h	SI	M7330AL5S1	M7330AL5S1_KWb	40030
ion7330_AL5_S1_KW c (Pc)	a	60h	SI	M7330AL5S1	M7330AL5S1_KWc	40032
ion7330_AL5_S1_KW tot (Ptot)	a	60h	SI	M7330AL5S1	M7330AL5S1_KWtot	40034
ion7330_AL5_S1_KVAR tot (Qtot)	a	60h	SI	M7330AL5S1	M7330AL5S1_KVARtot	40042
ion7330_AL5_S1_KVA tot (Stot)	a	60h	SI	M7330AL5S1	M7330AL5S1_KVAtot	40050
ion7330_AL5_S1_PFsign tot	a	60h	SI	M7330AL5S1	M7330AL5S1_PFsigntot	40058
ion7330_AL6_S1_Vln a	a	60h	SI	M7330AL6S1	M7330AL6S1_Vlna	40011
ion7330_AL6_S1_Vln b	a	60h	SI	M7330AL6S1	M7330AL6S1_Vlnb	40012
ion7330_AL6_S1_Vln c	a	60h	SI	M7330AL6S1	M7330AL6S1_Vlnc	40013
ion7330_AL6_S1_Vln avg	a	60h	SI	M7330AL6S1	M7330AL6S1_Vlnavg	40014
ion7330_AL6_S1_Vll ab	a	60h	SI	M7330AL6S1	M7330AL6S1_Vllab	40015
ion7330_AL6_S1_Vll bc	a	60h	SI	M7330AL6S1	M7330AL6S1_Vllbc	40016
ion7330_AL6_S1_Vll ca	a	60h	SI	M7330AL6S1	M7330AL6S1_Vllca	40017
ion7330_AL6_S1_Vll avg	a	60h	SI	M7330AL6S1	M7330AL6S1_Vllavg	40018
ion7330_AL6_S1_I a	a	60h	SI	M7330AL6S1	M7330AL6S1_Ia	40019
ion7330_AL6_S1_I b	a	60h	SI	M7330AL6S1	M7330AL6S1_Ib	40020
ion7330_AL6_S1_I c	a	60h	SI	M7330AL6S1	M7330AL6S1_Ic	40021
ion7330_AL6_S1_I avg	a	60h	SI	M7330AL6S1	M7330AL6S1_Iavg	40022
ion7330_AL6_S1_Freq	a	60h	SI	M7330AL6S1	M7330AL6S1_Freq	40025
ion7330_AL6_S1_KW a (Pa)	a	60h	SI	M7330AL6S1	M7330AL6S1_KWa	40028
ion7330_AL6_S1_KW b (Pb)	a	60h	SI	M7330AL6S1	M7330AL6S1_KWb	40030
ion7330_AL6_S1_KW c (Pc)	a	60h	SI	M7330AL6S1	M7330AL6S1_KWc	40032
ion7330_AL6_S1_KW tot (Ptot)	a	60h	SI	M7330AL6S1	M7330AL6S1_KWtot	40034
ion7330_AL6_S1_KVAR tot (Qtot)	a	60h	SI	M7330AL6S1	M7330AL6S1_KVARtot	40042
ion7330_AL6_S1_KVA tot (Stot)	a	60h	SI	M7330AL6S1	M7330AL6S1_KVAtot	40050
ion7330_AL6_S1_PFsign tot	a	60h	SI	M7330AL6S1	M7330AL6S1_PFsigntot	40058

Anexo 11. Archivo de Configuración renovar.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#                  RENOVADOR DE ACCIONES 1.0               #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa         #
# VersiÃ³n: 0000000001                                     #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

# Importa la librería de lectura/escritura de memorias
from src.lib.memoria_registros import *

# Adquiere los datos pendientes de...
#from src.etcbin.adquirir_modbus import *
#from src.etcbin.adquirir_ftp import *
#from src.etcbin.adquirir_sftp import *

# PESYR_FTP (no necesario)
#from src.etcbin.adquirir_pesyr import *
#memory_write(listado_telemandos,config.memoria())
# SIGNALS_FTP (no necesario)
#from src.etcbin.adquirir_signals_ftp import *
#memory_write(listado_telemandos,config.memoria())
# SIGNALS_WEB (no necesario)
#from src.etcbin.adquirir_web import *
#memory_write(listado_telemandos,config.memoria())
# Realimentador
#from src.etcbin.realimentacion import *
#memory_write(r.listlines(),config.memoria())
# Graficas
from src.etcbin.graficas import *
memory_write(graficas,config.memoria())
# Emails
from src.etcbin.emails import *
a.renovar()
```

Anexo 12. Archivo de Configuración cargar.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#                  ACCIONES DE CARGAR 1.0                  #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani               #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa        #
# VersiÃ³n: 0000000001                                    #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com       #
############################################################

# Cargo la configuración por defecto
from src.etcbin.cargador_sql import *
# Cargo la configuración por defecto
from src.etcbin.senales import *
# Restringe el listado de señales a solo las que se van a cargar en
la BD
s.filter(enBD=True)
# Cargo las señales en el objeto SQL
sql.senales(s)


# Argumentos {{{1
# Librería
import sys
# Argument: silent {{{2
debug=True
last=False
if (((len(sys.argv)==2) or (len(sys.argv)==3)) and (sys.argv[1]=='-
-silent')):
      debug=False
if ((len(sys.argv)==3) and (sys.argv[2]=='--silent')):
      debug=False
# }}}2
# }}}1

# Debug
sql.set_debug(debug)
# Cargar a la base de datos
sql.load()
```

Anexo 13. Archivo de Configuración /etc/graficas.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#              CONFIGURACION DE GRAFICAS v1.0              #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani                #
# Fecha: Riobamba, 15 de Enero del 2011                    #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa         #
# VersiÃ³n: 0000000001                                     #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                      #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com        #
############################################################

# Importo las señales disponibles
from graficas_default import *
from src.lib.fecha_actual import *

# Documentación {{{1
# Resolución (851) - Ancho de márgenes(70) - 1 = Ancho de la caja
de graficas
# La caja de graficas tiene N casillas, mas una en el extremo
derecho, mas una en el extremo izq (para 24 casillas de 1 horas son
26 casillas)
# Para calcular los pixeles que quedan para la graficas = size-1-
30(MargenD)-40(MargenI)
# }}}1
# Preconfig {{{1
# Colores
blanco=0
negro=1
rojo=2
verde=3
azul=4
azul_claro=5
magenta=6
amarillo=7
gris=8
verde_oscuro="darkgreen"

color_b1=azul
color_b2=rojo
color_b3="darkgreen"
color_b4="darkorange3"
color_b5= "darkcyan"
color_nivel=negro
color_presion="darkmagenta"
color_caudal=azul
color_frecuencia = "darkcyan"
color_potencia = "gray50"
color_cloro = rojo

# Forma de las líneas
```

```
escalera="escaleraPasado"
lineal="lineal"
evento="evento"

# Forma de los puntos (media, máximo, mínimo)
circuloV=1
triangulo_arribaV=2
cruz=3
equisV=4
romboV=5
triangulo_abajoV=6
cuadrado_equis=7
cruz_equis=8
rombo_cruz=9
circulo_cruz=10
triangulo_arriba_abajo=11
cuadrado_cruz=12
circulo_cruz=13
cuadrado_triangulo=14
cuadrado=15
circulo_mediano=16
triangulo=17
rombo=18
circulo_grande=19
circulo_pequeno=20
circuloC=21
cuadradoC=22
romboC=23
triangulo_arribaC=24
triangulo_abajoC=25
asterisco=26
punto=27
cero1=28
cero2=29
cero3=30
mas='"+"'
menos='"-"'
dospuntos='":"'
vertical='"|"'
porciento='"%"'
sostenido='"#"'


# Tamaño de los puntos (media, máximo, mínimo)
puntos_pequenios= 0.5
puntos_medianos = 1.3
puntos_grandes = 1.8

# Tipo de líneas
solida=1
puntos1=2
puntos2=3
puntos3=4

# Máximos y mínimos
maximos=(triangulo_arribaC,cuadradoC,True)
```

```
mínimos=(triangulo_abajoC,cuadradoC,True)
medias=(circuloC,False)
mediasUnica=(circuloC,True)

# Resolución
size=851
res_1pxminuto=1631
res_otra=1241

# Evento normal
evento_valor=[0,1]
evento_color1=("black",[None,negro],None)
evento_color0=("black",[negro,None],None)

#Evento color cobertura
#cobertura_valor=[0,32]
#digital_cobertura2=[]
#for i in range(0,100,5):
#    digital_cobertura2.append("gray%s" % (i))
#cobertura_color=("black",digital_cobertura2)

cobertura_valor=[0,12,16,32]
cobertura_color=("black",["black","gray50",None],None)

#digital_cobertura2=[]
#for i in range(0,100,5):
#    digital_cobertura2.append("gray%s" % (i))
#cobertura_color=("black",digital_cobertura2)


#Evento color Trafico de red
evento_valor_trafico_red=[0,1000,2200,4400,999999]
evento_trafico_red=("black",[None,"gray70","gray30","black"],None)

# Lista de graficas
graficas=[]
# }}}1

########################### GENERAL ###################################


f=datos_actuales()
valores=f.maxymin(30)
fecha=f.fecha_actual()
print "valores[0]: ", valores[0]
print "valores[1]: ", valores[1]

g=GRAFICA("Corriente a",s)
g.config(config)
g.leyendaColumnas(5)
g.sizey(valores[0],valores[1],1,1.5)
g.addaxis(0,"izq","Corriente Ia")
g.addaxis(10,"inf","Horas")
g.sizex(fecha[0],fecha[1],"horas","%H/%M/%S")
#g.sizex("date:ayer + 8h","date:hoy + 8h","horas")
```

```
#g.sizex("date:08/11/2010_08:00:00","date:09/11/2010_07:50:00","dia
s","%d/%m/%Y")

g.test("M7650T1_Ia",fecha[0],fecha[1],-10,5000)
#g.test("M7650T1_Ia","date:08/11/2010_08:00:00","date:09/11/2010_07
:50:00",-10,5000)

g.addline("ion7650 T1 Corriente Fase
A","M7650T1_Ia","a",escalera,color_caudal,1,solida)
g.leyenda("Temperatura Minima")

g.outpng("ion7650_T1_Corriente Fase A -
%iayer.png","ion7650_T1_Corriente Fase A - %ayer",res_otra,700)
graficas.append(g)
```

Anexo 14. Archivo de Configuración graficas.py

```
#!/usr/bin/python
#-*- coding: utf-8 -*-
############################################################
#                         GRAFICAS                         #
############################################################
# Autor: Francisco NarvÃ ez, Wilmer Guamani               #
# Fecha: Riobamba, 30 de Diciembre del 2010               #
# DescripciÃ³n: Instalador de Scada-Likindoy-Eersa        #
# VersiÃ³n: 0000000001                                    #
#                                                          #
# Codigo fuente bajo licencia GNU/GPL                     #
# francisco_jnq@hotmail.com, wilmer_g_p@hotmail.com       #
############################################################

# Importa la libreria de lectura/escritura de memorias
from src.lib.memoria_registros import *
# Pinto las grÃ¡ficas
from src.etcbin.graficas import *
# Librerias
import os
import sys


# Argumentos {{{1
# Argument: silent {{{2
# Importa la libreria de lectura/escritura de memorias
from src.lib.memoria_registros import *
# Importa la libreria de lectura/escritura de memorias
from src.lib.memoria_registros import *

if (((len(sys.argv)==2) or (len(sys.argv)==3)) and (sys.argv[1]=='--
silent')):
      debug=False
elif ((len(sys.argv)==3) and (sys.argv[2]=='--silent')):
      debug=False
else:
      debug=True
# }}}2
# Argument: last {{{2
if (((len(sys.argv)==2) or (len(sys.argv)==3)) and (sys.argv[1]=='--
last')):
      last=True
elif ((len(sys.argv)==3) and (sys.argv[2]=='--last')):
      last=True
else:
      last=False
# }}}2
# }}}1
# Cargo de la memoria la listas de graficas pendientes por hacer
pendientes=memory_read(graficas,config.memoria())
# Hay trabajo por hacer?

if (pendientes==[]):
      # Si no hay nada pendiente, indico que no hay nada que hacer
      lista=[]
```

```
else:
      # Si no es la ultima ejecucion
      if (not last):
            # Reviso si ya puedo generar todas las graficas
            todas=True
            for grafica in graficas:
                  if (not grafica.dotest()):
                        todas=False
                        break
            # Si no tengo todas las graficas, genero sÃ³lo las que
tengo pendientes en memoria
            if (not todas):
                  # Cargo de la memoria la listas de graficas
pendientes por hacer
                  lista=pendientes
            else:
                  # Si las tengo todas, pues las hago todas
                  lista=graficas
      else:
            # Si es la Ãºltima ejecuciÃ³n hacemos todas las que se
puedan
            lista=graficas

# Si hay algo que hacer
if (lista!=[]):
      # VacÃa la memoria temporalmente
      memory_write([],config.memoria())

      # Genera las grÃ¡ficas
      pendientes=[]
######################### aumentar codigo
      i=0
      for grafica in lista:
            grafica.set_debug(debug)
            try:
#                 print "path= ",paths[i], "pisicion= ",i
#                 time.sleep(3)

                  grafica.out(paths[i])
                  i=i+1
            except Exception,e:
                  # Extraigo el nombre del fichero que ha podido crear
R
                  fichero=grafica.fileR()
                  # Si dicho fichero existe
                  if ((fichero!=None) and (os.path.exists(fichero))):
                        # Elimino el fichero
                        os.unlink(fichero)
                  # Anoto el error
                  print "Error generando la graficas %s: %s\n" %
(grafica.nombre(),e)
                  pendientes.append(grafica)
                  i=i+1

      memory_write(pendientes,config.memoria())
```

BIBLIOGRAFÍA:

1. EMPRESA ELÉCTRICA RIOBAMBA S.A (EERSA): Información general.

   http://eersa.itgo.com

   201004

2. INTERNATIONAL BUSINESS MACHINES. Nueva York, Estados Unidos. Linux System Administration I TCP/IP and Services Certified: course material. 3a.ed. Nueva York, IBM, 2006. 100 p.

3. LENGUAJE DE PROGRAMACIÓN R

   http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_R

   http://www.r-project.org/

   201005

4. LIKINDOY: SCADA Libre.

   http://www.likindoy.org/es/

   201005

5. MEDIDOR ION 7330

   http://abampere.com.ar/pdf/ION7300_serie_folleto_largo_Espanol.pdf

   201007

6. MEDIDOR ION 7550/7650

http://www.monografias.com/trabajos11/sisco/sisco.shtml

http://abampere.com.ar/pdf/ION7550_7650_folleto_largo_Espanol%5B1%5D.pdf

201007

7. **PENIN, R. y AQUILINO.** Sistemas SCADA. 2da. ed. Barcelona, Marcombo. 2007. pp. 19-30.

8. PROTOCOLO Modbus TCP/IP

http://www.infoplc.net/Documentacion/Docu_Comunicacion/EthernetIndustrial/infoPLC_net_Ethernet_Industrial_ModbusTCP_IP.html

http://www.simplymodbus.ca/TCP.htm

201007

9. **SCHNEIDER ELECTRIC.** Edit. Canadá. Power Logic ION7300: Manual de Instalación, BC, V8M 2A5, Canadá, Schneider Electric, 2006. 30 p.

10. _____. Edit. Canadá. Power Logic ION7550/ION7650: Manual de Instalación, BC, V8M 2A5, Canadá, Schneider Electric, 2007. 32 p.

11. SUPERVISORY CONTROL AND DATA ADQUISITION (SCADA)

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

http://www.automatas.org/redes/scadas.htm

http://www.directindustry.es/prod/satec-ltd/software-scada-58438-380093.html

http://www.oocities.com/gabrielordonez_ve/SCADA.htm

201005

12. **TABOADA, J. y DENIS, J.** Proyecto Likindoy: Documentación, V. 20080703. 43 p. (Documento)

13. **TIPÁN, S.** Automatización e Integración al Sistema SCADA de los Alimentadores a1/1 y a1/2 de la Empresa Eléctrica Riobamba S.A. Tesis Ing. Electrico. Quito. Escuela Politecnica Nacional. Facultad de Ingenieria Electrica y Electronica. 2009. 149p.