REVISTA DIGITAL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
TELEM@TICA
Revista Telem@tica. Vol. 15. No. 7, enero-abril, 2016, p.80-92 ISSN 1729-3804

SISTEMA DE CÓDIGO ABIERTO PARA LA SUPERVISIÓN DE ALARMAS: OPENSIGER.

MS.C Denis Morejón López[1], MS.C Leonel Dorta Gómez[2], Ing. Yurisdan López Villa[3]

[1] Empresa de Telecomunicaciones de Cuba S.A, Cuba, Calle 33 Edif. E Apto 7, Pastorita, Cienfuegos.
[2] Empresa de Telecomunicaciones de Cuba S.A, Cuba, Marcial Gómez # 1255 Norte Final, Ciego de Ávila.
[3] Empresa de Telecomunicaciones de Cuba S.A, Cuba, Edif. 6 Apto 23, Buenavista, Cienfuegos.
[1] denis.morejon@etecsa.cu

RESUMEN

OpenSiger es un sistema para obtener, procesar, almacenar y mostrar vía web alarmas de una red de autómatas programables de la marca Omron C200HX que está montada actualmente en las plantas telefónicas analógicas que persisten en ETECSA, aunque pudiera tener otros usos futuros. Las alarmas son estados críticos de variables como la temperatura del equipamiento, el nivel del tanque de combustible para el grupo electrógeno, fallas de corriente alterna, etc. Es un reemplazo con tecnologías de código abierto de un sistema, de código cerrado y sin soporte técnico, nombrado SARE, pero conocido popularmente en la empresa como Siger. El sistema SARE (o Siger) presentaba problemas de estabilidad y soporte para nuevas funcionalidades. Son esos los motivos por los que mediante la búsqueda de documentos técnicos de esa fecha y técnicas de ingeniería inversa se obtuvo el protocolo que existía para la comunicación del SARE con los autómatas y así programarlo en el nuevo sistema. También se cambió el lenguaje de programación (ahora Python), el sistema operativo (ahora Linux), la tarjeta de adquisición de datos (ahora se pueden usar adaptadores USB - Serie), la interfaz gráfica (ahora web), el gestor de base de datos (ahora PostgreSQL), y en alguna medida la estructura de la propia base para ajustarse a las nuevas necesidades.

Palabras Clave: autómata, alarmas, código abierto.

ABSTRACT

OpenSiger is a system to collect, process, store and display web alarms of programmable controllers (PLCs) from Omron C200HX brand via network, currently mounted on analog PBXs that persist in ETECSA. This system could have other uses in the future. Alarms are critical states of variables such as temperature of the equipment, the level of fuel tank in the generator, AC failure, etc. OpenSiger is a replacement with open source technologies for a system named SARE, but popularly known in the company as Siger. The SARE (or Siger) system had problems of stability and support for new features, right now SARE is an unsupported system. Those are the reasons we obtained to develop the new system seeking technical documents and doing reverse engineer to the protocol existed for

communication with PLCs and SARE.

The graphical interface (now web), the programming language (now Python), the operating system (now Linux), the data acquisition board (Series can now use USB adapters) and also the database (now PostgreSQL) changed, and to some extent the database structure itself to meet the new requirements.

Keywords: programmable controllers, alarms, open source.

INTRODUCCIÓN

Por el año 2000 se comenzó a implementar un sistema nombrado SARE cuyo objetivo era supervisar el estado de las variables alrededor de una planta telefónica de tecnología analógica. Se adquirieron en el extranjero autómatas programables de la marca Omron y el modelo C200HX. El objetivo del autómata era el de capturar eventos o alarmas en la instalación de la planta, desde un problema del funcionamiento de la propia planta hasta un evento de puerta abierta, o temperatura fuera de rango.

El sistema concebía un autómata por planta. Luego esos autómatas se comunicaban por puerto serie, a través de módems con líneas dedicadas, con una PC colectora central de todos los datos de las centrales. Después esa PC colectora se comunicaba con otra que tenía una base de datos MSSQL donde se almacenaban los datos permanentemente.

Los supervisores accedían a las alertas a través de una tercera PC cliente, con un programa de escritorio concebido para acceder a las alertas activas en la base de datos.

Este programa fue encargado a un grupo de desarrollo de la Universidad Central de la Villas (UCLV) que ya no existe, y que vendieron el producto sin el código fuente del mismo.

Como consecuencia el colector de datos, programa encargado de interactuar con el autómata, que constituía el eslabón principal del sistema, era una especie de caja negra para todo el que deseara programar mejoras a un sistema que quedó sin atención ni soporte técnico del proveedor.

Tras los primeros años de explotación el sistema comenzó a comportarse inestable, a dejar de señalizar alarmas, o a dejar puestas en pantalla otras que ya se habían restablecido. Los supervisores de conmutación enviaban en ocasiones a técnicos a las centrales remotas cuando no hacía falta, o enviaban tarde a otros cuando las alarmas reales llevaban tiempo presentes. Si no ocurrió un problema grave en medio de la falta de fiabilidad en las alarmas fue por pura casualidad y porque los supervisores llamaban a los administradores de red, muchas veces en horario extra-laboral, para que reiniciaran el sistema y así validar la veracidad de las alarmas.

Se tiene entonces como problema científico que el sistema para la supervisión de plantas analógicas SARE presenta un alto grado de inestabilidad y esto provoca que no sean fiables sus alarmas. Como consecuencia se envían técnicos a las localidades donde están las plantas para solucionar averías inexistentes, o se envía a los administradores de red a reiniciar los servidores del sistema para validar la veracidad de dichas alarmas.

Se define como objeto de estudio el sistema para la supervisión de plantas analógicas SARE y como campo de acción la supervisión de plantas analógicas en la dirección territorial de ETECSA en Cienfuegos en el período 2014-2015.

Como objetivo general se tiene el programar un nuevo sistema para la supervisión de plantas analógicas utilizando tecnologías de código abierto que reutilicen la red de autómatas instalada en la dirección territorial de ETECSA en Cienfuegos, y mejore la estabilidad y fiabilidad de dicho proceso.

Se definieron los objetivos específicos siguientes:

- Analizar la base de datos, la estructura general y el funcionamiento del sistema SARE como base para la concepción de un nuevo sistema.
- Identificar los mensajes que se intercambian en el protocolo existente entre el servidor colector de alarmas y los autómatas programables Omron C200HX.
- Instalar variantes de hardware más actuales para la tarjeta multipuertos de adquisición de datos que posee el servidor colector de alarmas.
- Diseñar el colector de datos, la base de datos, y la interfaz de usuario del nuevo sistema teniendo en cuenta el estudio realizado al sistema anterior.
- Desarrollar la programación del OpenSiger teniendo en cuenta el paradigma Modelo, Vista, Controlador (MVC).
- Validar los resultados del trabajo.

Y por todo lo antes expuesto se llega a la hipótesis de que, crear un sistema completamente nuevo para la supervisión de plantas analógicas, que reemplace al actual sistema SARE puede ser costoso en términos de horas/hombres dedicadas a una tarea compleja, pero es la mejor y más radical solución al problema de estabilidad que presenta.

CONTENIDO

A continuación se detallan los principales conceptos y tecnologías utilizadas.

- ✓ **Central Telefónica**

La central telefónica es el punto donde se reúnen las conexiones de todos los aparatos telefónicos de una determinada área, que se denomina “área local” o “área central”. La central que efectúa únicamente la misión de conectar abonados entre sí, se denomina central local. En ella reside la inteligencia necesaria para encaminar correctamente la llamada desde su origen (abonado llamante), hasta su destino (abonado llamado).

Al conjunto de los elementos necesarios para unir una central local con sus abonados, se denomina “red de abonados” o “red local” de la central.

Las centrales telefónicas o centrales de conmutación son las encargadas de proporcionar las funciones para poder realizar una llamada, de las cuales, la má importante es la de “conexión” o “conmutación” de los abonados llamante y llamado. El componente principal de una central de conmutación es el “equipo de conmutación”, compuesto por una serie de órganos automáticos y de circuitos.

El conjunto de órganos y circuitos que forman el equipo de conmutación se divide en dos partes: red de

conexión y unidad de control. La red de conexión comprende el conjunto de órganos y circuitos, que constituyen el soporte físico de la comunicación. Por lo tanto, es a la red de conexión de la central, donde se conectan las líneas de abonado y los enlaces.

La red de conexión está constituida por un número muy elevado de circuitos. En una red de conexión puede haber hasta tres tipos de etapas: Concentración, Distribución y Expansión. Atendiendo al tipo de señal eléctrica que conmuta, las redes de conexión se dividen en Analógicas y Digitales. Una red de conexión analógica conmuta señales analógicas, y una red de conexión digital conmuta señales digitales. Una señal analógica es aquella que puede variar de forma continua, es decir, tomando un número ilimitado de valores distintos, y una señal digital es aquella que sólo puede tomar un cierto número de valores, es decir, varía de una forma discreta. La señal digital más utilizada es la señal digital binaria que sólo puede tomar dos valores, denominados "0" lógico y "1" lógico.

- ✓ **Redes de conexión analógica espacial**

Por un mismo camino físico de la red de conexión, sólo puede establecerse una única comunicación. Ya que si dos comunicaciones se establecieran por el mismo camino físico, se sumarían las dos señales analógicas correspondientes. Lo que diferencia a una comunicación de otra distinta en el interior de una red de este tipo, es el hecho de que discurran por caminos físicos distintos, separados en el espacio. De ahí, que a la red de conexión se la llame analógica-espacial.

- ✓ **Autómata**

En electrónica un autómata es un sistema secuencial, aunque en ocasiones la palabra es utilizada también para referirse a un robot. Puede definirse como un equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales. Sin embargo, la rápida evolución de los autómatas hace que esta definición no esté cerrada.

- ✓ **Autómata programable**

Un autómata programable se puede considerar como un sistema basado en un microprocesador, siendo sus partes fundamentales la Unidad Central de Proceso (CPU), la Memoria y el Sistema de Entradas y Salidas (E/S).

La CPU realiza el control interno y externo del autómata y la interpretación de las instrucciones del programa. A partir de las instrucciones almacenadas en la memoria y de los datos que recibe de las entradas, genera las señales de las salidas. La memoria se divide en dos bloques, la memoria de solo lectura o ROM (Read Only Memory) y la memoria de lectura y escritura o RAM (Random Access Memory).

En la memoria ROM se almacenan programas para el correcto funcionamiento del sistema, como el programa de comprobación de la puesta en marcha y el programa de exploración de la memoria RAM.

La memoria RAM a su vez puede dividirse en dos áreas:

- Memoria de datos, en la que se almacena la información de los estados de las entradas y salidas

y de variables internas.

- Memoria de usuario, en la que se almacena el programa con el que trabajará el autómata.

El sistema de Entradas y Salidas recoge la información del proceso controlado (Entradas) y envía las acciones de control del mismo (salidas). Los dispositivos de entrada pueden ser pulsadores, interruptores, finales de carrera, termostatos, presostatos, detectores de nivel, detectores de proximidad, contactos auxiliares, etc.

Por su parte, los dispositivos de salida son también muy variados: Pilotos indicadores, relés, contactores, arrancadores de motores, válvulas, etc. En el siguiente punto se trata con más detalle este sistema.

Para controlar un determinado proceso, el autómata realiza sus tareas de acuerdo con una serie de sentencias o instrucciones establecidas en un programa. Dichas instrucciones deberán haber sido escritas con anterioridad por el usuario en un lenguaje comprensible para la CPU. En general, las instrucciones pueden ser de funciones lógicas, de tiempo, de cuenta, aritméticas, de espera, de salto, de comparación, de comunicación y auxiliares.

- ✓ **Python**

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,1 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

Python usa tipado dinámico y conteo de referencias para la administración de memoria. Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

- ✓ **PostgreSQL**

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de Código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Algunas de sus principales características son, entre otras: la alta concurrencia. Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que se añadió o también conocido por lo que

se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. Amplia variedad de tipos nativos PostgreSQL permite crear una amplia funcionalidad a través de su sistema de activación de disparadores (triggers).

- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.
- Soporte para transacciones distribuidas. Permite a PostgreSQL integrase en un sistema distribuido formado por varios recursos

Algunos de los lenguajes que se pueden usar son los siguientes:

- Un lenguaje propio llamado PL/PgSQL (similar al PL/SQL de Oracle).
- C.
- C++.
- JavaPL/Java web.
- PL/Perl.
- plPHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme.
- Lenguaje para aplicaciones estadísticas R por medio de PL/R.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta (query en inglés).

Las funciones pueden ser definidas para ejecutarse con los derechos del usuario ejecutor o con los derechos de un usuario previamente definido. El concepto de funciones, en otros DBMS, son muchas veces referidas como "procedimientos almacenados" (stored procedures en inglés).

- ✓ **Bootstrap**

Bootstrap, es un framework de código abierto, una colección gratuita y de código abierto de herramientas para la creación de sitios web y aplicaciones web. Contiene plantillas CSS de diseño para tipografía, formas, botones, navegación y otros componentes de la interfaz, así como las extensiones de JavaScript opcionales. El marco de arranque tiene como objetivo facilitar el desarrollo web.
Bootstrap constituye el front-end, es decir, una interfaz para el usuario, a diferencia del código del lado del servidor que reside en el "back-end" o servidor.

Es compatible con las últimas versiones de los navegadores más usados como Google Chrome, Firefox, Internet Explorer, Opera y Safari, aunque algunos de estos navegadores no son compatibles con todas las plataformas.
Desde la versión 2.0 también es compatible con el diseño web responsivo. Esto significa que el diseño de

las páginas web se ajusta dinámicamente, teniendo en cuenta las características del dispositivo utilizado (de escritorio, tableta o teléfono móvil).
Desde la versión 3.0, Bootstrap adoptó una primera filosofía de diseño móvil, haciendo hincapié en el diseño de respuesta o responsivo por defecto.

A continuación se explicarán los distintos componentes de software del programa y la relación entre ellos.

- ✓ opensiger_daemon.py

El dominio opensiger es un proceso que corre permanentemente y es el encargado de consultar a los autómatas en busca de posibles alarmas detectadas. Se trata de un ciclo indefinido abriendo todos los puertos serie conectados a los autómatas para comunicarse con éstos. La figura 1 muestra una idea resumida.

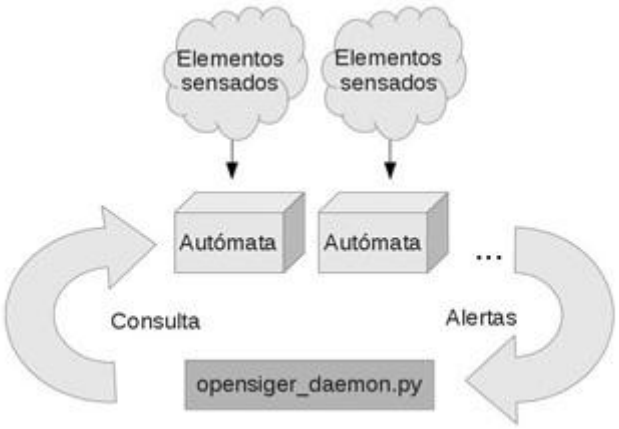

Fig.1 Ciclo de consultas a los autómatas por parte del dominio opensiger_daemon.py

Este dominio es controlado por un servicio ubicado en el directorio /etc/init.d/ y nombrado opensiger. El servicio se controla como se hace típicamente en un entorno Linux:

/etc/init.d/opensigerstart (Para arrancar el servicio de captura de datos)

/etc/init.d/opensiger stop (Para detener el servicio)

/etc/init.d/opensiger status (Para ver el estado del servicio)

El opensiger_daemon.py es configurado a través de un fichero nombrado opensiger.conf que se carga en memoria cada vez que se reinicia el servicio. El demonio deja las trazas de operaciones fallidas en el fichero opensiger.log. Sitúa las alarmas en la base de datos a través de un fichero especializado en consultas nombrado postgres.py.

- ✓ protocol.py

opensiger_daemon.py utiliza el fichero protocol.py donde está definido el protocolo o forma de comunicarse con los autómatas. A su vez, protocol.py emplea clases y métodos definidos en el fichero plc_client.py.

- ✓ main.py

Otra pieza clave en el sistema es la cgi nombrada main.py. Esta cgi controla el acceso, a través del servidor web apache, a la interfaz web del sistema. Utiliza tanto componentes html prediseñados como acceso a la base de datos mediante el componente postgres.py antes mencionado

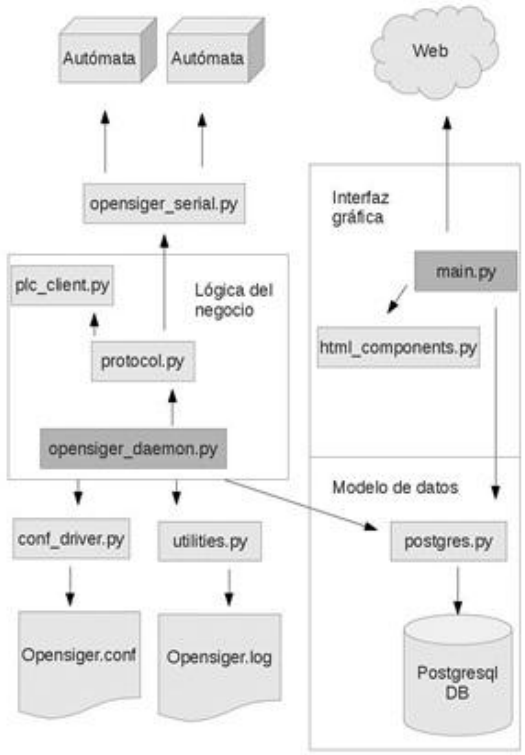

Fig.2 Esquema general del sistema OpenSiger.

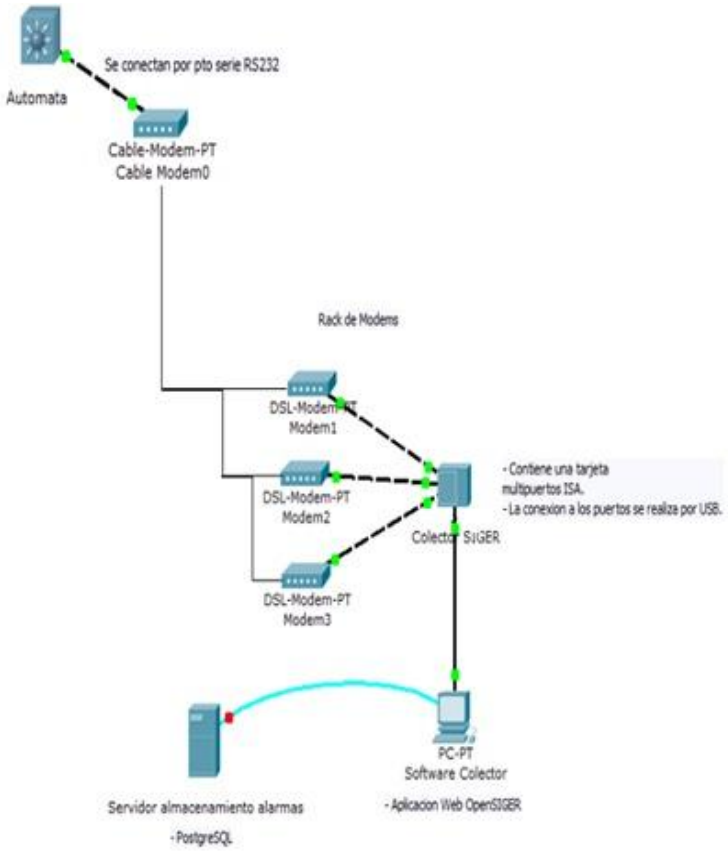

Fig.3 Diagrama de dispositivos del nuevo sistema OpenSiger.

Resultados

La siguiente figura muestra la comparación del comportamiento de las incidencias reportadas al Centro de Dirección Territorial (CDT) referentes a problemas con el Siger, el que estaba antiguamente en el 2014 y con el nuevo OpenSiger en lo que va de 2015, cabe resaltar que los valores corresponden a iguales períodos, o sea de Enero a Junio en ambos años.

Fig.4 Problemas reportados durante el período 2014 - 2015.

Además de recurrir al lugar que canaliza los problemas cuando los especialistas y técnicos llaman para reportar los problemas, el Centro de Dirección Territorial, se realizó una encuesta tomando como población/muestra a 10 personas, entre ellos Jefes de Departamento y especialistas que trabajan en el Centro de Gestión del territorio, las personas que más necesitan del software desarrollado por estar incluido en su contenido de trabajo. Los datos estadísticos fueron procesados con el software de código abierto PSPP (alternativa opensource al SPSS de Windows).

Por los datos censados: se constató mediante la encuesta que el 80 % de los encuestados confía más en las alarmas que brinda el OpenSiger que el antiguo Siger, se comprobó además que el 90 % de los encuestados coincide en que la disponibilidad del OpenSiger con respecto al Siger es mucho mayor. Un 70 % de los encuestados piensa que el OpenSiger y su interfaz brindan más información que el Siger y que debido a ello les es más fácil realizar la toma de decisiones.

Lo más destacado de la encuesta fue poder comprobar que el 100 % de los encuestados opina que fue conveniente el cambio de un sistema por el otro y las mejoras introducidas tanto por software como por hardware.

Valoración Técnico - Económica

A la hora de hacer una valoración económica del desarrollo del sistema OpenSiger se tienen que tener en cuenta varios aspectos en los que se contemplen gastos con relación a si permaneciera en explotación el anterior sistema Siger y que inciden en el criterio final. Los dos primeros aspectos son gastos que se incrementan en el tiempo, y el tercer aspecto es un gasto fijo por la compra a tercero de una solución al problema.

Gastos de combustible y horas/hombres para atender una falsa interrupción.

Supongamos que aparece una interrupción en la planta nombrada "5 de Septiembre", perteneciente al municipio Aguada de Pasajeros. También vamos a asumir que es del tipo de interrupción en la que debe personarse un técnico de Cienfuegos para normalizarla.

Estas variables tienen valores aproximados:

D (Distancia_Cienfuegos_Aguada) = 60km
KpL (Km_por_litro_auto_ligero) = 14
P (Precio_litro_combustible) = 1 CUC
I (Número de interrupciones por mes) = 1
M (Muestra de tiempo) = 6 Meses

Gasto_combustible = Litros_gastados * Precio
Litros_gastados = I*M (2 * D / KpL)
Gasto_combustible = I*M (2 * D / KpL) * P
Gasto_combustible = 6*(2 * 60 / 14) * 1
Gasto_combustible = **51.6 CUC**

Pérdidas asociadas a dejar de atender una interrupción importante por considerarla falsa.

Supongamos que en lugar de atender una falsa interrupción dejamos de atender una que sí es verdadera e implica la interrupción de una planta analógica (como Antonio Sánchez) por un período de 24 horas.

IM (Ingreso_mensual_promedio) = 19397 CUP
ID (Ingreso_diario) = IM / 30 = 647 CUP
P (Pérdidas en 24 horas) = ID = 647 CUP

Costos en contratar a un tercero la programación del nuevo producto de gestión.

Supongamos que contratamos con un tercero el diseño, programación, e implementación del sistema nuevo. Una empresa nacional de desarrollo de software y servicios informáticos como Desoft cobra la hora/hombre a 11,84 CUC aproximadamente.
Este sistema fue desarrollado por 3 personas en aproximadamente 6 meses de trabajo.

Costo_Hora_Hombre = 11,84 CUC
Hombres = 3
Total_Horas = 6 * 192h = 1152h
Horas_Hombres = Hombres * Total_horas
Horas_Hombres = 3456
Costo_con_tercero = Horas_Hombres * Costo_Hora_Hombre
Costo_con_tercero = 40919,04CUC

A este costo hipotético habría que restarle el gasto en horas/hombres en el que se incurriópor el pago aproximado que nos hace la empresa en CUC.
Costo_Hora_Hombre = 2 CUC
Gasto_salarial = 3456 * 2 = 6912 CUC

Haciendo la resta para sacar el costo neto:
Costo_neto = Costo_con_tercero - Gasto_salarial
Costo_neto = **34007,04CUC**

Resumiendo

Vamos a considerar como valor económico el monto ahorrado en no tener que adquirir un software con una empresa de programación como Desoft, que hubiera sido de aproximadamente **34007 CUC**.

CONCLUSIONES

1. Se realizó un análisis de todos los componentes del anterior sistema Siger y esto facilitó la comprensión de su modo de operar, para así diseñar el nuevo sistema introduciendo mejoras en su funcionamiento.

2. Se sustituyó la tarjeta multipuertos serie de adquisición de datos, basada en bus ISA, por varios adaptadores USB-SERIE que pueden ser usados en computadoras modernas. Esta adaptación permitió utilizar un servidor profesional con prestaciones más modernas que favorecen la estabilidad del servicio.

3. Se programó e implementó el sistema teniendo en cuenta características del anterior y basado en el paradigma MVC.

4. Finalmente se realizó una encuesta de aceptación, y se obtuvieron reportes del centro de dirección territorial donde se evidencian la mejora en la estabilidad y disponibilidad del proceso de supervisión de plantas analógicas en la DTCF.

REFERENCIAS BIBLIOGRÁFICAS

- PROGRAMMABLE CONTROLLERS C200HX/C200HG/C200HE. OPERATION MANUAL OMRON. JANUARY 1998.
- GARCÍA ALGARRA, JAVIER (2012). "DE GRAN VÍA AL DISTRITO C. EL PATRIMONIO ARQUITECTÓNICO DE TELEFÓNICA"
- KNOWLTON, JIM (2009). PYTHON. TR: FERNÁNDEZ VÉLEZ, MARÍA JESÚS (1 EDICIÓN). ANAYA MULTIMEDIA-ANAYA INTERACTIVA. ISBN 978-84-415-2513-9.
- MARTELLI, ALEX (2007). PYTHON. GUÍA DE REFERENCIA. TR: GORJÓN SALVADOR, BRUNO (1 EDICIÓN). ANAYA MULTIMEDIA-ANAYA INTERACTIVA. ISBN 978-84-415-2317-3.

- PÁGINA WEB: HTTP://WWW.POSTGRESQL.ORG/
- PÁGINA WEB: HTTP://WWW.POSTGRESQL.ORG/SUPPORT/PROFESSIONAL_SUPPORT

- PÁGINA WEB: HTTP://BLOG.GETBOOTSTRAP.COM/