Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Unidad de hardware para un sistema de control supervisor y de adquisición de datos empotrado utilizando herramientas de software libre

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Isaías Sancho Cordero

Cartago, 20 de enero de 2012

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. Arys Carrasquilla Batista

Profesor lector

Ing. Anibal Coto Cortés

Profesor lector

Ing. William Marín Moreno

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 20 de enero de 2012

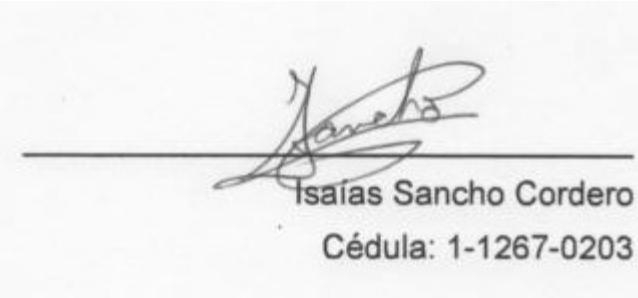
Declaratoria de autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo realizado y por el contenido del correspondiente anteproyecto.

Cartago, 20 de enero de 2012.



Isaías Sancho Cordero

Cédula: 1-1267-0203

Resumen

La empresa SIESA desarrolla e implementa sistemas SCADA (Supervisory Control and Data Accuisition, control supervisor y adquisición de datos) para distintas industrias a nivel nacional; sin embargo, los sistemas SCADA que utiliza son de carácter propietario de alto costo y no pueden modificarse para casos específicos de trabajo. Además SIESA se vuelve solo un intermediario para comercializarlos y no percibe una ganancia por el software sino solo por su implementación.

A la empresa le interesa tener un software de carácter propio para poder modificarlo y personalizarlo y que además le aporte una ganancia mayor en la instalación de sistemas SCADA industriales. También importa poder comercializar los sistemas de control automático hacia otros sectores comerciales y de hogar donde las soluciones existentes son muy caras. Por esto se necesita un sistema SCADA propio, de bajo costo de implementación que pueda ser mantenido y personalizado para cada caso específico y que trabaje en sistemas de bajo consumo de recursos y energía.

En este documento se presenta el diseño de una unidad controladora con características de hardware empotrado que esté conectada a la red local o por Internet y que permita acceder a la interfaz de usuario desde cualquier computadora por medio de dicha red.

Esta unidad de control se desarrolla bajo licencias de software libre para poder continuar su desarrollo en el futuro y crear una comunidad de usuarios alrededor de ésta que funcionen como depuradores y desarrolladores del sistema.

Abstract

SIESA Company develops and implements SCADA (Supervisory Control and Data Accuisition) systems for a variety of industries nationwide. However, the SCADA systems they use are licensed and expensive and can not be modified for specific implementations. In addition SIESA becomes only an intermediary to sell those softwares and do not receive any profit from the software but only for the implementation of it.

The company is interested on having a software of its own for edition and customization and that also brings them a greater gain on installation of industrial SCADA systems. They are also interested on selling automatic control systems to other sectors like business and home where the existing solutions are too expensive. That's why they need a SCADA system of their own, with a low implementation cost that can be maintained and customized for each specific case and that works in low-power systems and energy resources.

The aim is to design a control unit featured on embedded hardware that is connected to the network either local or Internet that allows access to the user interface from any computer via the network.

This control unit is developed under free software licenses to continue its development in the future and to create a user community around it, working as free testers and developers for the system.

Agradecimientos

Quiero darle gracias a Dios por permitirme estar aquí cerrando esta etapa de mi vida y por darme sabiduría y paciencia para lograr mis metas.

Agradezco también a mis padres María Paulina Cordero y Rodolfo Sancho por apoyarme siempre, por creer en mi y por enseñarme que las cosas se logran con trabajo y dedicación.

A mis hermanos, mi familia y mis amigos que me han ayudado con su pequeño granito de arena en todos mis éxitos.

Índice

Índice de figuras

Índice del tablas

Capítulo 1. Introducción

El presente trabajo se realizó como parte del proyecto SCADA en la empresa ClearSoft S.A. en consorcio con la empresa SIESA. Este proyecto surge de la necesidad de SIESA de contar con un sistema se control supervisor y adquisición de datos carácter propio y de código abierto, que sea económico, escalable, configurable y mantenible con facilidad.

El proyecto SCADA tiene como objetivo obtener un sistema de control supervisor y adquisición de datos que reúna las cualidades de programas similares pero sin tener las desventajas que presentan estos y que pueda competir con otros sistemas del mismo ámbito en el mercado. Un sistema así brinda una ventaja competitiva a SIESA, pues puede diseñar e implementar sistemas de control industrial a un menor costo y con un programa que puede ser configurado y revisado para adaptarse a situaciones y problemas nuevos.

También se busca crear una comunidad de desarrolladores y depuradores como en otros proyectos de software libre que ayude al sistema a mejorar mucho más rápido al incluir ideas y mejoras de muchas personas.

El consorcio entre las empresas SIESA y ClearSoft se crea como beneficio mutuo, pues la primera cuenta con el conocimiento requerido en el campo de la electromecánica y la segunda tiene varios años especializada en diseño de soluciones en computación y en sistemas de código abierto para todo tipo de industrias.

1.1. Problemática del proyecto SCADA

Los sistemas SCADA comerciales que existen en la actualidad tienen un costo elevado para cualquier aplicación, por lo que las implementaciones de este tipo solo son accesibles para empresas con suficientes recursos y una necesidad real de estos sistemas. Entonces, una empresa pequeña o que cuente con un bajo presupuesto no puede invertir en un sistema de supervisión y control para sus necesidades, lo que deja fuera del alcance del mercado a una gran parte de los posibles clientes. También la utilización de estos sistemas para soluciones en domótica queda prácticamente excluido del panorama.

Las soluciones comerciales también presentan el problema de no ser manipulables y personalizables hasta los detalles más específicos de cada implementación, pues se hacen para cumplir con muchos casos generales pero no con un caso específico para cada cliente. Por ejemplo, cuando se trabaja con sistemas de gran tamaño y con muchas variables, los programas comerciales se vuelven muy costosos, pues trabajan con un número limitado de variables y para sobrepasar ese límite se debe adquirir nuevas licencias. Entonces un desarrollador como SIESA se encuentra con limitantes de implementación y costo

económico para los casos que trabaja y debe adaptar la solución de cada problema al programa que utiliza pues no puede modificar el programa para adaptarse a la solución diseñada.

Por otro lado, las soluciones de carácter libre (open source) que ya existen se encuentran en dos categorías, algunas se han creado para aplicaciones muy específicas y no son fáciles de acoplar a implementaciones de mayor complejidad y más generales, y otras son soluciones que se encuentran en fase de desarrollo y no tienen todavía un producto final que sea confiable y seguro.

También, las soluciones que existen en la actualidad funcionan sobre una computadora que debe estar encendida 24 horas para su acceso en el momento en que se necesite, lo que representa un consumo de energía elevado. Además, al tratarse de programas de terceros, la empresa SIESA se vuelve solamente un intermediario en la venta de este tipo de sistemas y no percibe una ganancia considerable al utilizar soluciones de carácter comercial.

Se presenta en SIESA una necesidad de desarrollar un sistema SCADA propio, de bajo costo de implementación que pueda ser mantenido y personalizado para cada caso específico y que trabaje en sistemas de bajo consumo de recursos y energía.

1.2. Unidad controladora de hardware

Tal y como se aprecia en la figura 1, un sistema SCADA puede dividirse en tres componentes principales, aunque por lo general su implementación suele estar en una sola unidad de hardware. La primera parte es la base de datos, encargada de guardar de forma ordenada los datos de supervisión y control que utiliza y adquiere el sistema. En segundo lugar está la unidad de control que es la encargada de supervisar y controlar cada dispositivo o grupo de dispositivos. Por último el programa SCADA es la cabeza desde dónde el usuario puede administrar todo el sistema, controlar los actuadores y guardar y leer desde la base de datos.

Se proyecta diseñar e implementar una unidad de control en un sistema empotrado independiente del sistema central, que permita la adquisición de datos y el manejo de los dispositivos de control. Esta unidad debe comunicarse con el sistema central utilizando la tecnología Ethernet de manera local o remota.

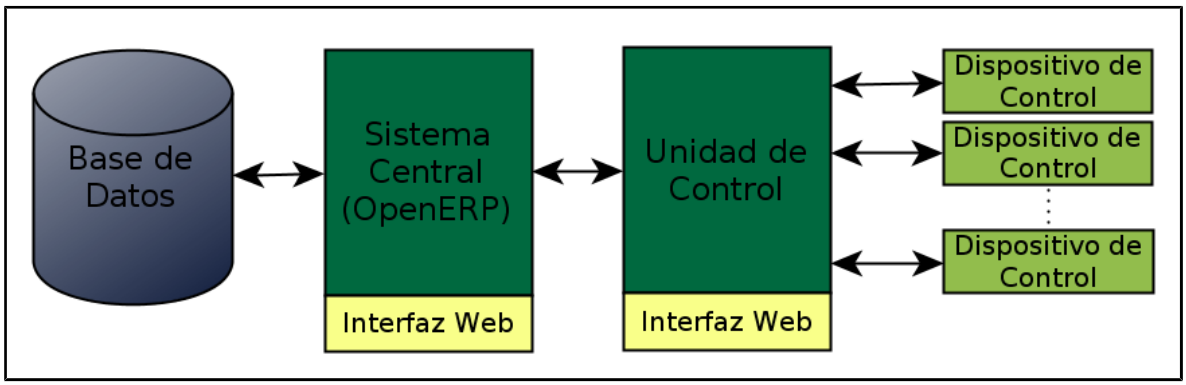


Figura 1. Diagrama general de un sistema SCADA

El usuario podrá controlar todo el sistema desde una interfaz web, para permitir una supervisión desde cualquier máquina en la red y no solo desde el servidor principal o una computadora específica. Para probar el sistema, éste se instalará en una casa de habitación, donde ya se encuentra un sistema comercial en funcionamiento el cual controla toda la iluminación, un calentador de agua solar, un módulo de presión constante de agua potable, un sistema de riego, un sistema de una piscina y otros sistemas menores.

La meta del proyecto SCADA es implementar un sistema de control supervisor y adquisición de datos que pueda competir con las soluciones comerciales del mercado. Para esto se deben considerar diferentes características que den a este sistema la competitividad deseada. El sistema debe entonces cumplir con los siguientes requisitos:

- *Económico*: El costo de implementación del sistema no debe de ninguna forma superar el costo de implementación de una solución comercial ya existente que es de cerca de $ 7500.
- *Flexible*: El sistema debe adaptarse fácilmente y con modificaciones simples a cualquier caso de control que se requiera monitorizar.
- *Eficiente*: Se debe hacer un uso óptimo de un hardware y software con bajo consumo de recursos y energía.
- *Abierto*: El software debe ser de carácter libre para que se pueda mejorar y seguir desarrollando en el futuro.

Capítulo 2. Meta y objetivos.

2.1. Meta

Construir una solución para sistemas de control supervisor y adquisición de datos que sea económica, flexible y eficiente a nivel de software y hardware, que sea de código abierto que pueda sustituir las soluciones comerciales que se utilizan actualmente.

2.2. Objetivo general

Diseñar e implementar una unidad electrónica ("puller") que permita la supervisión, el control y la adquisición de datos de dispositivos de control industrial, que sea económica, y flexible, que se comunique con el servidor central SCADA por medio de Ethernet.

2.3. Objetivos específicos

- Desarrollar un controlador ("puller") que trabaje en un sistema computacional embebido basado en GNU/Linux, que sea compacto y de bajo consumo de recursos y energía. Se debe obtener un consumo de energía para el controlador menor a 100 W.
- Implementar una comunicación con los dispositivos de control que sea eficiente y permita controlar estos dispositivos y además adquirir datos y supervisar la actividad. El tiempo de respuesta de los actuadores debe ser menor a 500 ms.
- Obtener un canal de comunicación en el que se refresque al menos 200 variables por segundo, para mantener consistencia entre los datos y configuraciones del "puller" y de la base de datos aún cuando haya cortes de comunicación.
- Implementar una interfaz gráfica web que permita al administrador configurar el "puller" de manera remota sin necesidad del servidor central y que además le permita controlar y observar el funcionamiento del sistema que se va a monitorizar en todo momento.
- Desarrollar un sistema completo cuyo costo de producción no sobrepase el costo del sistema que se va a reemplazar, el cual es de $7517.

Capítulo 3. Marco teórico

En este capítulo se desarrollan de manera general los siguientes temas:

- *Sistemas SCADA*: Fundamentos y aplicaciones de este tipo de sistemas.
- *Sistemas empotrados*: Nociones importantes del diseño de componentes de hardware y software para soluciones de funciones dedicadas.
- *Protocolo de comunicación Modbus*: Aplicaciones y uso de este protocolo en sistemas de control.
- *Bases de datos*: Fundamentos y aplicaciones de sistemas de almacenamiento de datos.

3.1. Sistemas SCADA

SCADA es un acrónimo en inglés de *Supervisory Control And Data Acquisition* (control supervisor y adquisición de datos.). Como lo indica su nombre, no se trata de un sistema de control completo, pero se enfoca más bien en la supervisión de diferentes aparatos con los que interactúa, como controladores lógicos programables (PLCs) u otros módulos de hardware comerciales.

Los sistemas SCADA se utilizan comúnmente para la supervisión de procesos industriales, donde se tiene desde unos 1000 canales de entrada y salida (E/S) hasta un millón o más canales.

La arquitectura de un SCADA se puede separar en dos capas, la capa del cliente donde se tiene una interfaz humano máquina personalizable y la del servidor de datos, que maneja el control y procesamiento de datos del sistema [1]. Un modelo de esta arquitectura se puede observar en la figura 2, donde se aprecia los módulos de los servidores de datos, que se encargan de manejarlos controladores, y los módulos clientes separados.

A nivel de software, se usan sistemas multitarea localizados en uno o varios servidores que son los responsables de la adquisición de datos de los controladores, el manejo de alarmas y control. En general se utiliza un solo computador central para todo el sistema, pero hay casos particulares en que se tiene varios servidores dedicados para tareas específicas y un maestro que los administra [1].

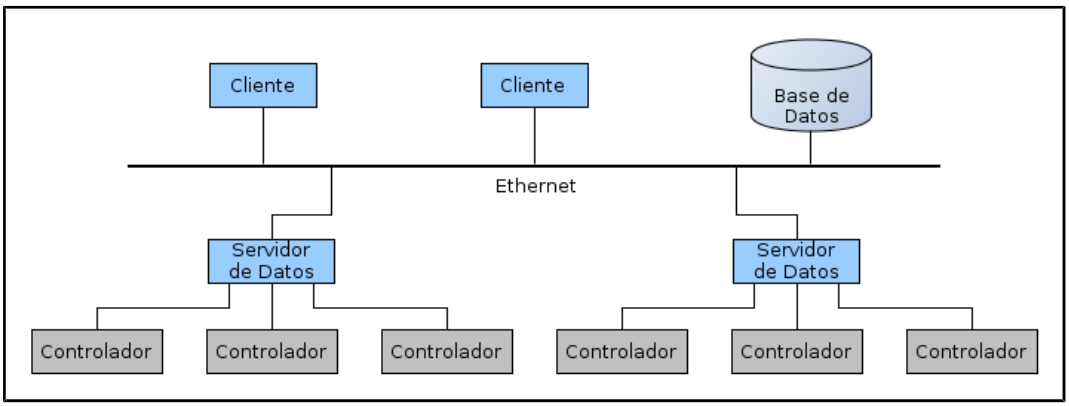


Figura 2. Arquitectura típica de hardware de un sistema SCADA.

La HMI (Human Machine Interface, interfaz humano máquina) corresponde al programa con el que el usuario puede interactuar para ver y controlar el sistema. Está conectada a la base de datos del sistema SCADA y permite monitorizar procesos, programar momentos de mantenimiento de diferentes máquinas y controlar funciones de la planta que se esté supervisando.

Una de las más importantes funciones de esta interfaz es el manejo de alarmas. Cuando el sistema detecta un estado de alarma, este puede proceder a realizar diferentes acciones, como encender los indicadores de alarma para que un operario se alerte y corrija el problema, o incluso modificar algunos parámetros de control para salir de la condición de alarma [1].

3.2. Sistemas empotrados

Un sistema empotrado (también llamado sistema embebido, incrustado o integrado) es un sistema basado en un microprocesador y construido para realizar una función específica o un rango de funciones dedicadas. Se diferencia de una computadora de uso general porque el usuario final tiene control sobre la operación del aparato pero no puede cambiar las funciones que realiza [2].

Otra característica que diferencia un sistema empotrado de una computadora de uso general es su bajo costo y bajo uso de recursos. Los dispositivos integrados se utilizan para reducir el gasto de implementación para las tareas específicas que se les da, por lo que suelen tener procesadores de menos de 1 GHz y memorias reducidas para disminuir los costos. Estos sistemas suelen estar implementados en una sola placa madre, donde se encuentra el microprocesador, la memoria y los distintos periféricos de entradas y salidas que se le integren.

Los sistemas empotrados se caracterizan por estar ajustados para funcionar en tareas específicas, muchas veces con sistemas de tiempo real, incluso muchos de estos elementos solo realizan una tarea específica. Aunque la mayoría de los sistemas funcionan con programas directamente en ensamblador, se suele utilizar

también sistemas operativos para facilitar la implementación de tareas y programas en diferentes lenguajes y con ayuda de otras herramientas.

Los sistemas operativos que se utilizan suelen ser también reducidos en cuanto a consumo de recursos para adaptarse al hardware que se tiene, pero brindan control sobre el sistema de forma eficiente para los programas que se requieran.

3.3. Protocolo de comunicación MODBUS

MODBUS es un protocolo de comunicación ubicado en la capa de aplicación, capa siete del sistema OSI, que proporciona comunicación tipo cliente/servidor entre varios dispositivos conectados a diferentes tipos de buses y redes.

El estándar para comunicaciones MODBUS por puerto serie existe desde 1979 y se sigue utilizando hoy en día para muchas aplicaciones industriales de comunicación y control. Su simplicidad hace que este protocolo siga en expansión e incluso se utilice actualmente para comunicaciones por TCP/IP [3].

3.3.1. Descripción del protocolo MODBUS

En la figura 3 se aprecia la trama del protocolo MODBUS, donde se define una unidad de datos de protocolo (*Protocol Data Unit* o PDU en inglés) simple e independiente de las capas superiores de comunicación. Cuando se trabaja en un tipo de bus o red específica se pueden introducir nuevos campos en la unidad de datos de aplicación(*Application Data Unit* o ADU en inglés).

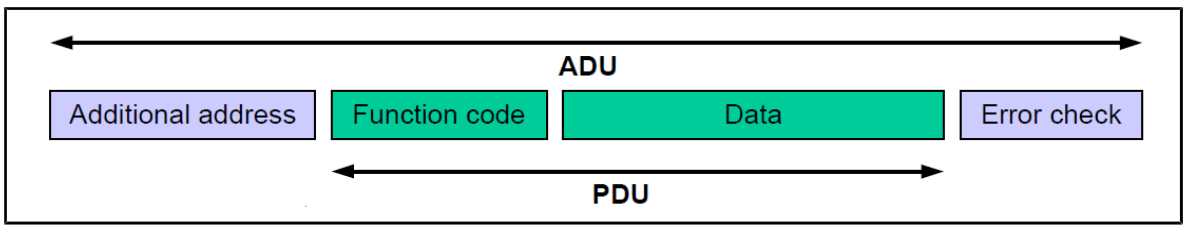


Figura 3. Trama general del protocolo MODBUS [3].

La comunicación cliente/servidor sobre protocolo MODBUS es iniciada por el cliente, que construye la trama de datos. La función es la que indica al servidor que acción debe tomar y la trama de datos le da al servidor diferentes parámetros para ejecutar esta función.

El código de función es de un byte y está predefinido por el estándar, sus valores van de 1 a 255, estando los valores 128 a 255 reservados para usarlos en excepciones. Este código es el que utiliza el cliente para avisar al servidor sobre la acción que debe realizar.

El campo de datos se utiliza para brindar una mayor información al servidor sobre la función que debe realizar. Si la acción del servidor se realiza con éxito, este responde al cliente de la misma manera y utiliza la trama de datos para regresar la información que pidiera el cliente.

En caso de ocurrir una excepción o error en el servidor, este devuelve el mismo byte de función que recibió del cliente, pero cambiando el bit más significativo por un uno lógico. En este caso el campo de datos lleva información sobre el error que se produjo [3].

3.4. Bases de datos

Una base de datos es una estructura computacional que almacena un conjunto de datos útiles para el usuario final y una serie de metadatos. Los metadatos son las características que poseen los datos y las relaciones que existen entre estos datos en la base de datos [4].

Existen muchos tipos de bases de datos diferentes que se pueden clasificar en seis modelos, sin restringirlas a estos modelos únicamente:

- bases de datos tradicionales,
- bases de datos jerárquicas,
- bases de datos en red,
- bases de datos relacionales,
- bases de datos orientadas a objetos y
- bases de datos declarativas [5].

3.4.1. Bases de datos relacionales

El modelo relacional fue postulado por Edgar Frank Codd en 1970 basado en la lógica de predicados y en la teoría de conjuntos. Está fundamentado en el uso de relaciones o tuplas de datos.

La base de datos relacional es aquella que utiliza relaciones para guardar los datos. Está compuesta por varias tablas, en las filas de las tablas se representan los registros de datos. Para relacionar los datos se utilizan claves primarias y foráneas [4].

La clave primaria corresponde al campo que se utiliza como identificador de un registro. Esta clave debe ser única y no puede contener valores NULL. La clave foránea es un campo que guarda la referencia a una clave primaria de otra tabla. Las claves foráneas no necesitan ser únicas pues muchos registros pueden tener una referencia hacia un mismo registro padre.

Las bases de datos se deben organizar separando dos secciones importantes: el esquema y los datos.

El esquema es la definición del orden de la base de datos y almacena los nombres de las tablas junto con las columnas que le pertenecen a cada tabla. Además se almacenan los nombres y tipo de datos de cada columna. En el caso de las bases de datos relacionales, este esquema debe ser normalizado para permitir que la base de datos sea utilizada de una manera óptima

Los datos, también llamados instancia, corresponden al contenido de una o varias tablas en un momento específico, pero también se puede hablar de instancia cuando se muestra un subconjunto de datos contenidos en una relación.

Capítulo 4. Consideraciones previas de diseño

En el mercado existen varios productos y sistemas SCADA propietarios con características particulares que pueden ser utilizados en distintas soluciones industriales. Algunos de estos sistemas con los que SIESA ya ha trabajado son IntegraXor [6], Indusoft [7] u Opus [8]. Sin embargo estos productos suelen tener un costo elevado para cualquier aplicación, por lo que las implementaciones de este tipo solo son accesibles para empresas con suficientes recursos y una necesidad real de estos sistemas. En este sentido, la empresa pequeña que cuenta con un bajo presupuesto no puede invertir en un sistema de supervisión y control para sus necesidades, lo que deja fuera del alcance del mercado a una gran parte de los posibles clientes. También la utilización de estos sistemas para soluciones personales en casas de habitación queda prácticamente excluido del panorama.

Además los sistemas SCADA comerciales están diseñados para cumplir con las especificaciones comunes de la mayoría de implementaciones que se realizan, por lo que otro gran problema es que no se puede manipular ninguna sección de código del programa para corregir errores o personalizar el sistema para casos específicos. El sistema que se compra no se puede mejorar para situaciones especiales de una implementación. Esto lleva a tener que utilizar un sistema comercial diferente según las especificaciones de cada implementación y además se debe adaptar los casos especiales para que el programa SCADA que se utilice pueda manejarlos.

También existen algunas soluciones de carácter libre (código abierto), pero estas se encuentran en dos categorías, existen algunas que fueron diseñadas para una implementación muy específica y no son fáciles de adaptar a cualquier sistema más general, como Likindoy [9]; las otras son sistemas SCADA que se encuentran en fase de desarrollo y no tienen todavía un producto final que sea confiable y seguro como FreeSCADA [10], ARGOS [11] u OpenSCADA [12].

Un detalle importante sobre los sistemas SCADA existentes es que deben funcionar en una computadora o servidor que debe estar encendido sin interrupciones para su acceso en cualquier momento y para estar recopilando datos de forma fiable. Esto representa un consumo de energía elevado para el sistema.

Se vislumbra entonces una necesidad de diseñar un nuevo sistema SCADA tomando en cuenta las ventajas y características del *software* que ya existe pero mejorando los puntos en los que estos sistemas fallan. Se quiere entonces que el nuevo diseño sea de bajo costo de implementación que pueda ser mantenido y personalizado para cada caso específico, que trabaje en sistemas de bajo consumo de recursos y energía y que sea capaz de trabajar en soluciones para empresas pequeñas pero que pueda ser escalado hasta grandes magnitudes que algunas empresas necesitan.

Un sistema SCADA puede dividirse en dos partes importantes, la supervisión de los dispositivos de control y el manejo de datos. Aunque la mayoría de sistemas en el mercado cuentan con una sola unidad administrativa que realiza estas dos funciones, para la implementación particular se decidió separar estas dos partes en diferentes dispositivos. Esto permite que el manejo y presentación de los datos se haga desde un sistema central que sea robusto y potente en cuanto a análisis y registro. Por otro lado, la supervisión se delega a las unidades de control. Estos son sistemas empotrados, de bajo consumo de recursos capaces de vigilar varios dispositivos de control a la vez y conectados vía Internet a un solo sistema central.

Una configuración así permite la escalabilidad de cualquier implementación hasta grandes series de sistemas industriales en diferentes localidades geográficas con un solo núcleo central de manejo de datos. Además al utilizar dispositivos empotrados para la supervisión se tiene un dispositivo económico y de bajo consumo de recursos que proporciona un ahorro de energía y la posibilidad de trabajar por periodos de tiempo sin necesidad de tener activo el servidor central.

Para el presente proyecto de graduación se desarrolló el diseño e implementación de la unidad de control y supervisión. Esta unidad de control forma parte del proyecto SCADA y por eso se toma en cuenta también el núcleo central, encargado de la administración de datos. Sin embargo el diseño del núcleo central está fuera del alcance de este informe.

Se debe considerar algunos aspectos importantes del núcleo central, éste está basado en el programa OpenERP, que corresponde a un programa de planificación de recursos empresariales de carácter libre y que permite la incorporación de módulos específicos para aplicaciones que el usuario defina. Se utiliza este programa pues esta implementado de forma muy robusta y modular, tiene un manejo muy eficiente de bases de datos y una interfaz muy amigable y configurable para el usuario. Además este sistema por si solo ya tiene la capacidad de trabajar en computadoras de bajo consumo y en la nube. El servidor central corresponde a una serie de módulos del OpenERP que recopilan los datos de las diferentes unidades de control y los ordenan en la base de datos para consultas y manipulación.

A continuación se definen los aspectos para el diseño de la unidad controladora del sistema SCADA que se presenta en este informe. Entre las características fundamentales se debe notar:

1. **Respecto al *hardware* del sistema:** Para asegurar una implementación de bajo consumo de recursos y económica se debe contar con las siguientes características:
    - Tarjeta de hardware de bajos recursos y bajo consumo de energía, pensada para sistemas empotrados.
    - Puerto Ethernet para la comunicación con la red.

- Al menos cuatro puertos USB para expansiones.
- Puerto serie RS-232 para comunicación directa por consola.

2. **Respecto al *software* del sistema:** Para asegurar un sistema robusto y eficiente que además pueda trabajar separado del núcleo central se debe considerar un *software* con las siguientes características:
    - Manejo de una base de datos para guardar su estado y configuración en caso de cortes de comunicación con el sistema central.
    - Manejo de históricos de datos para proporcionarlos al sistema central cuando se necesite.
    - Comunicación eficiente con los dispositivos de control para mantener un estado de cada dispositivo constantemente.
    - Manejo adecuado de las instrucciones que se envían a los dispositivos.
    - Interfaz de reconfiguración del sistema sin necesidad de pasar por el núcleo de datos.
    - Interfaz de control y supervisión en caso de no tener acceso al sistema central.
    - Desarrollo completo bajo licencias libres y con herramientas de *software* libre.

4.1. Síntesis de la solución

La solución desarrollada trata de un sistema empotrado capaz de administrar varios controladores lógicos programables (PLC por sus siglas en inglés) y de manejar una comunicación constante de datos con el sistema central.

La unidad controladora es un sistema empotrado en una placa de hardware de bajos recursos por lo que se seleccionó el sistema operativo Voyage Linux, un sistema operativo libre, derivado de la distribución Debian de Linux y adaptado para sistemas de muy pocos recursos de memoria y procesador [13]. Se seleccionó este sistema para trabajar en un entorno similar al del sistema Ubuntu, también basado en Debian, que es el sistema en que trabaja el OpenERP que utiliza el servidor central, para tener un entorno de trabajo similar y poder en un futuro proporcionar una solución sencilla donde se pueda integrar el núcleo central y la unidad de control en una misma computadora en caso de que fuera necesario.

Dado que el sistema OpenERP está programado en el lenguaje de programación Python y que los módulos que se programaron también están en

este lenguaje, se prefirió utilizar el mismo lenguaje para la unidad controladora. Esto permite tener un solo lenguaje de programación para el proyecto SCADA y una mayor mantenibilidad de la solución completa.

El lenguaje de programación Python es un lenguaje de alto nivel, interpretado y multiparadigma, ya que soporta orientación a objetos, programación imperativa y programación funcional. Su filosofía se basa en una sintaxis muy limpia que favorece la legibilidad del código [14].

Para lograr un sistema óptimo se inició el proceso de diseño con la selección del *hardware* apropiado para la unidad de control. Este sistema debe tener la capacidad de manejar el sistema operativo Voyage Linux y mantener el perfil de dispositivo empotrado que se requiere para la solución.

4.1.1. Selección del sistema de hardware

Para esta selección se tomaron en cuenta tres posibles dispositivos, dos de ellos pensados para sistemas empotrados y uno que corresponde al sistema computacional más barato y pequeño en el mercado. La tabla 1 muestra una comparación de estos tres dispositivos.

La placa ALIX 6E1 de PC Engines y la Net5501-70 de Soekris Engeneering son soluciones con procesador, memoria y otros elementos ya integrados, mientras que el Atom mini-ITX Board de Intel es más bien una tarjeta madre de computadora a la que hay que agregarle una fuente de poder y memoria por separado para que funcione. Por esta razón la opción de Intel queda descartada pues no proporciona un sistema pequeño y empotrado. Sin embargo se utiliza para tener un punto de comparación con las otras dos alternativas.

Es importante conocer las capacidades de procesamiento y memoria que presenta cada sistema. En este rubro se observa que si bien la solución de PC Engines como la de Soekris Engeneering utilizan el mismo procesador AMD, diseñado para sistemas empotrados, la tarjeta Net5501-70 viene como el doble de memoria RAM que la ALIX 6E1, lo que podría ser ventajoso si el sistema llegara a necesitarlo. En cuanto al almacenamiento, En los dos casos se utilizaría una tarjeta de memoria CompactFlash, sin embargo la Net5501-70 permite utilizar también un disco SATA de mayor almacenamiento si llegara a necesitarse.

En comparación, la solución de Intel posee un procesador mucho más rápido y una capacidad de memoria considerablemente mejor. Lo que hace notar que un sistema computacional permitiría un desarrollo mucho más pesado. Esto se consideraría si en un futuro se llevara el proyecto a sistemas mucho más potentes y sin posibilidad de ser empotrados.

Tabla 1. Comparación de diferentes sistemas que se pueden utilizar como base para implementar la unidad de control.

	ALIX 6E1	Net5501-70	Atom mini-ITX Board
Fabricante	PC Engines	Soekris Engeneering	Intel
Procesador	AMD Geode LX800 (500 MHz)	AMD Geode LX800 (500 MHz)	Intel Atom (1,6 MHz)
Memoria RAM	256 MB (integrada)	512 MB (integrada)	Hasta 2 GB (no integrada)
Puertos para memoria de almacenamiento	CompactFlash	CompactFlash, SATA	IDE, SATA
Puertos integrados	RJ-45, USB 2.0, RS-232	RJ-45, USB 2.0, RS-232	PS/2, RS-232, VGA, USB 2.0, RJ-45, Audio, Puerto Paralelo
Puertos de expansión	MiniPCI, MiniPCI Express	PCI, MiniPCI	PCI
Tipo de alimentación	Adaptador de 12V, PoE	Adaptador de 12V	Fuente de poder ATX
Consumo de energía	5 W, máximo 12 W	5 W, máximo 15W	25 W
Precio de la placa base publicado en la tienda ($)	85	262	210
Precio aproximado con accesorios e impuestos. ($)	195	380	400

Los puertos de comunicación y puertos de expansión que manejan las dos soluciones integradas son muy similares y proporcionan las mismas ventajas sin importar el sistema. Cabe notar que cualquiera de los dos sistemas tienen una consola por puerto serial pues no poseen conexión a un monitor que de una salida visual para el usuario. El sistema Intel, por su parte si posee salidas para monitor y para otros periféricos, pues está pensada para trabajar como una computadora multifuncional.

El consumo de energía que especifica el fabricante es similar para los dos sistemas empotrados y es bastante bajo. En este punto no hay una diferencia comparativa importante.

En cuanto al precio en el mercado, la placa ALIX 6E1 tiene un costo de $195, lo que abarata la implementación de la unidad de control, mientras que el sistema de Soekris Engeneering tiene un precio apenas menor que el de una solución computacional más completa. Por esta razón, se decide utilizar el hardware de PC Engines aunque la memoria RAM sea menor, ya que sigue siendo suficiente para el sistema a desarrollar.

4.1.2. Evaluación del núcleo del OpenERP como base de desarrollo

Dentro del diseño original del sistema se pretendía desarrollar todo desde cero, empezando con un controlador central que trabajara como un *daemon* de Linux, un programa que corre en segundo plano en el sistema y que no es controlado por el usuario, y que permitiera el acople de módulos para manejar los distintos protocolos de comunicación que se puedan agregar en el futuro y algunas otras características y tareas que se quisieran agregar. Sin embargo, al trabajar con el sistema OpenERP para el desarrollo del núcleo central, en paralelo con el desarrollo de la unidad de control, se propuso utilizar el mismo núcleo de OpenERP llamado OpenObject, pues este cuenta con varias características que se pretendía diseñar y además está hecho de forma muy robusta y general.

Es importante notar que el OpenObject fue desarrollado con una arquitectura cliente/servidor. La comunicación entre estos se realiza por medio del protocolo XML-RPC en una red IP. Esto facilita el manejo del sistema desde un cliente, que en este caso es un cliente web, que puede no estar necesariamente en la solución empotrada, sino que puede estar instalado en cualquier otra máquina y acceder al sistema integrado desde cualquier lugar de la red [15].

El cliente como tal está pensado para no realizar prácticamente ninguna acción y que sea más bien el servidor el que maneje cualquier transacción necesaria, mientras que el cliente solo muestra datos al usuario de diferentes formas.

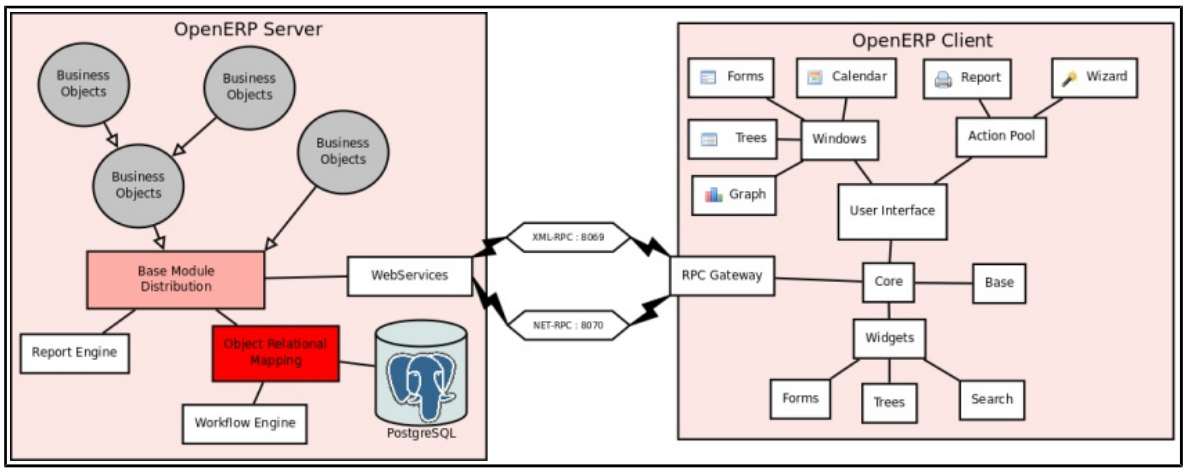


Figura 4. Arquitectura cliente/servidor que utiliza el sistema OpenObject [15]

Otro aspecto importante es el manejo que tiene OpenERP de la base de datos, que si bien no se pretende utilizar de forma exhaustiva por el alto consumo de memoria que puede tener, si se utiliza para guardar los datos de configuración del sistema, necesarios al arrancar y configurar la unidad de control.

El OpenERP también tiene un manejo de servicios web robusto, que usa principalmente para comunicación con el cliente, pero que se puede utilizar de la misma forma para la comunicación con el sistema central.

Cabe notar que el sistema OpenERP puede instalarse y utilizarse en ambientes de poco uso de recursos y que si no se inicia nada más que el núcleo base, éste puede correr en un sistema empotrado como el de la unidad de control que se desarrolla. Sin embargo se debe controlar de cerca el uso de la base de datos y de la cantidad de instancias que puede utilizar en paralelo, para evitar que el sistema se sature por la ejecución del OpenERP.

Tomando en cuenta estos puntos, se vislumbra el ahorro de tiempo de desarrollo que trae consigo la utilización del OpenObject, núcleo de OpenERP. Además se mantiene un desarrollo de todo el proyecto con las mismas herramientas tanto para el servidor central del SCADA como para la unidad de control. Esto asegura la compatibilidad entre todos los sistemas.

Se desarrolla entonces la implementación de la unidad de control como un módulo del sistema OpenERP. Este módulo se diseñó procurando mantener el uso de memoria y recursos al mínimo posible, para asegurar un funcionamiento óptimo del sistema.

Capítulo 5. Descripción detallada de la unidad de control

Se puede describir el diseño de la unidad de control como una estructura jerárquica en la que cada nivel corresponde a una capa de hardware o software necesaria, interconectada con las adyacentes pero independiente en el diseño.

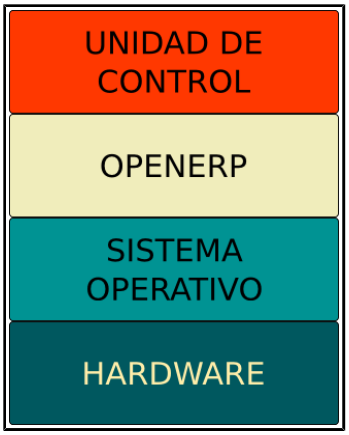


Figura 5. Diagrama jerárquico de la unidad de control.

El hardware es la parte física del sistema que se desarrolla. Corresponde a una tarjeta integrada de bajos recursos ALIX 6E1 de PC Engines. Sobre esta placa se instala el sistema operativo Voyage Linux, un sistema basado en Debian y adaptado para sistemas empotrados.

Tal y como se aprecia en la figura 5, en la capa siguiente se tiene el programa OpenERP, del cual solo se utiliza el núcleo. Este núcleo es el que permite manejar la base de datos y los servicios web. Además, la unidad de control está programada como un módulo de este sistema.

La capa de la unidad de control toma en cuenta el módulo de control SCADA que se diseñó para el OpenERP y todos los sistemas adyacentes que se necesitaron para que éste módulo trabajara como se requería.

5.1. Aspectos de hardware del sistema

La unidad de hardware que se utiliza es una placa ALIX 6E1, fabricada por PC Engines [16]. Este sistema cuenta con:

- Procesador AMD Geode LX800 de 500MHz.

- Memoria RAM de 256 MB.
- Tarjeta de memoria CompactFlash de 1 GB agregada.
- Un puerto de expanción miniPCI y otro miniPCI Express.
- Dos canales Ethernet.
- Un puerto serie RS-232 para la consola de eventos.
- Un segundo puerto RS-232 agregado a través de un convertidor USB-RS232 marca DeckCell.
- Dos puertos USB.

Se debe notar que este sistema tiene una capacidad de almacenamiento muy reducida, muy poca memoria RAM y un procesador muy lento comparado con cualquier computadora de propósito general de la actualidad. Por esta razón se debe procurar mantener el consumo de recursos del sistema al mínimo para no saturarlo y perder la supervisión sobre los controladores que se manejan.

5.2. Aspectos del sistema operativo

En la tarjeta ALIX 6E1 se instala el sistema operativo Voyage Linux. Este es un sistema derivado de Debian Squeeze y está pensado para funcionar de forma óptima en plataformas empotradas como las de PC Engines, Soekris y otras basadas en procesadores Atom de Intel [13].

La instalación básica de este sistema operativo necesita solamente 128 MB de espacio en disco, lo que hace que sea un sistema operativo muy reducido. Sin embargo, utiliza el sistema de paquetes de Debian, lo que permite instalar desde los repositorios muchos programas y bibliotecas que se requiera para un uso específico.

La versión de Voyage Linux que se utiliza es la 0.7.0, pero se modificó la estructura del sistema de archivos para usar *ext3* en lugar del *ext2* recomendado para este sistema operativo, pues el *ext3* tiene registro por diario (*journaling*), que permite restablecer los datos afectados por una transacción en caso de que esta falle, por ejemplo en cortes de corriente o por corrupción del sistema.

La instalación del sistema operativo es sencilla y viene con instrucciones. Sin embargo, se debe configurar para utilizarlo para la unidad de control. Esta configuración es tediosa y muy repetitiva, por lo que se desarrolló un *script* en bash que realiza esta configuración paso a paso con supervisión del usuario. La configuración cambia varios aspectos del sistema que se listan a continuación:

- Cambio de la contraseña del usuario *root*.
- Actualización del sistema.

- Instalación y configuración de *locales*, parámetros que definen el idioma, país y otras preferencias regionales del sistema.
- Configuración del nombre de dominio y dirección IP para el sistema.
- Configuración del huso horario.
- Sincronización de la hora con NTP.
- Instalación de Webmin.
- Otros cambios menores para el arranque del sistema.

Estas configuraciones permiten preparar el sistema para instalar en OpenERP, que va a funcionar como base de la unidad de control. También se procura tener acceso al dispositivo desde Internet por protocolo ssh, para asistencia remota.

5.3. Aspectos del programa OpenERP

Anteriormente se mencionó que se utiliza el núcleo del sistema OpenERP como programa de control. Esto se decidió para aprovechar el manejo que tiene OpenERP de la base de datos y de servicios web para comunicación.

La ventaja significativa que tiene el OpenObject es que se puede programar toda la unidad de control como un módulo agregado que se acopla al sistema y lo utiliza a su favor.

Los módulos de OpenERP funcionan como agregados al programa principal. Según los agregados que se instalen o no, se puede tener configuraciones muy distintas del sistema. Sin embargo existe un solo módulo que es necesario para el funcionamiento del sistema, éste es el módulo base. Este agregado puede verse como parte del núcleo del programa y es el que permite manejar el resto de módulos del sistema. También se encarga de definir algunos de los elementos básicos del OpenERP. Ningún otro de los módulos típicos del OpenERP se conservó porque no se necesita ninguna funcionalidad de manejo empresarial y de recursos para la unidad de control. Sin embargo, en caso de necesitarlo para una implementación específica, se procuró mantener la unidad de control compatible con el sistema típico.

Para la instalación del OpenERP se recurrió a un *script* desarrollado por ClearCorp, el cual permite instalar el OpenObject server, junco con todos los agregados, incluyendo algunos que son específicos para Costa Rica y de ClearCorp; y el OpenERP web, que corresponde al cliente con interfaz web para el sistema OpenERP.

Este programa de instalación se tuvo que modificar para que la instalación se adaptara al sistema Voyage Linux y a los bajos recursos de hardware. Lo primero fue quitar de la instalación el cliente web y todos los módulos. Esto permite tener

un servidor OpenObject básico. Luego se tuvo que modificar la instalación para que automáticamente configurara el servidor. El *script* utilizado solo instala en sistema, pero se tiene un segundo programa que configura el servidor. Esto se hace para poder configurar muchos servidores de pruebas en las máquinas de desarrollo sin tener que reinstalar el OpenERP cada vez. Sin embargo, en este caso se tiene un espacio reducido, por lo que se instala y configura directamente un solo servidor de OpenERP, sin dejar copias a modo de plantilla para nuevos servidores.

Por último también se modificó el *script* para agregar directamente el módulo de la unidad de control y el módulo base del sistema SCADA para el manejo de objetos de la base de datos. De esta forma se obtiene un servidor completamente funcional al terminar la instalación.

Algunas otras modificaciones se hicieron para instalar ciertos paquetes de *software* que el sistema Voyage no trae por defecto.

El programa final realiza una serie de instalaciones y configuraciones que se listan a continuación:

- Agregar los usuarios para el OpenERP.
- Instalar Python y las librerías requeridas para el servidor.
- Instalar bazaar para el control de versiones del desarrollo.
- Instalar PostgreSQL y configurarlo para usarlo como manejador de la base de datos.
- Descargar e instalar el OpenObject server y los agregados que corresponden a la unidad de control.
- Configurar los archivos de OpenERP con los valores dados para el servidor.
- Instalar y configurar apache.

La unidad de control es totalmente funcional luego de esta instalación. Lo único que falta es la creación de una base de datos y la conexión al servidor central.

La base de datos debe crearse manualmente desde una interfaz web del OpenERP que puede estar instalada en una computadora personal. Sin embargo no se debe llenar con ningún dato, pues esto lo hace la unidad de control al iniciar por primera vez.

5.4. Aspectos de la unidad de control

A continuación se detallan los aspectos importantes de la unidad de control, que incluye algunas características del sistema OpenERP y el funcionamiento del

módulo que se desarrolló para el control del sistema SCADA. También se incluye el funcionamiento de la interfaz web que se agregó por medio de Webmin.

La figura 6 muestra un diagrama de la arquitectura de la unidad de control. En ella aparecen los subsistemas más importantes, como la comunicación con los PLC, la conexión con la base de datos y la puerta de enlace para el sistema central y para la interfaz web.

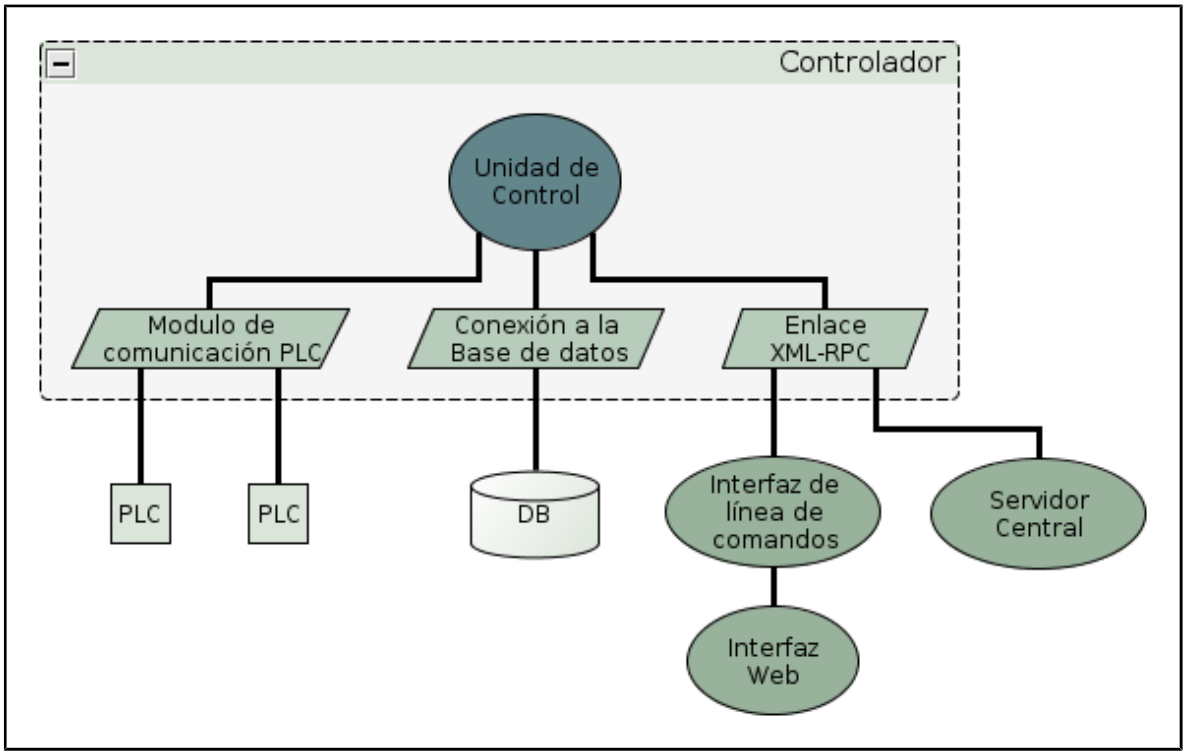


Figura 6. Diagrama de la arquitectura de la unidad de control.

Se debe notar que cada subsistema no es funcional por si solo, sino que es el núcleo de la unidad de control el que utiliza estos subsistemas según sea necesario.

El módulo de la unidad de control tiene una rutina de trabajo que corresponde al diagrama de flujo de la figura 7. Esta rutina está implementada como un subproceso del sistema OpenERP y trabaja en paralelo con todos los demás subprocesos del programa.

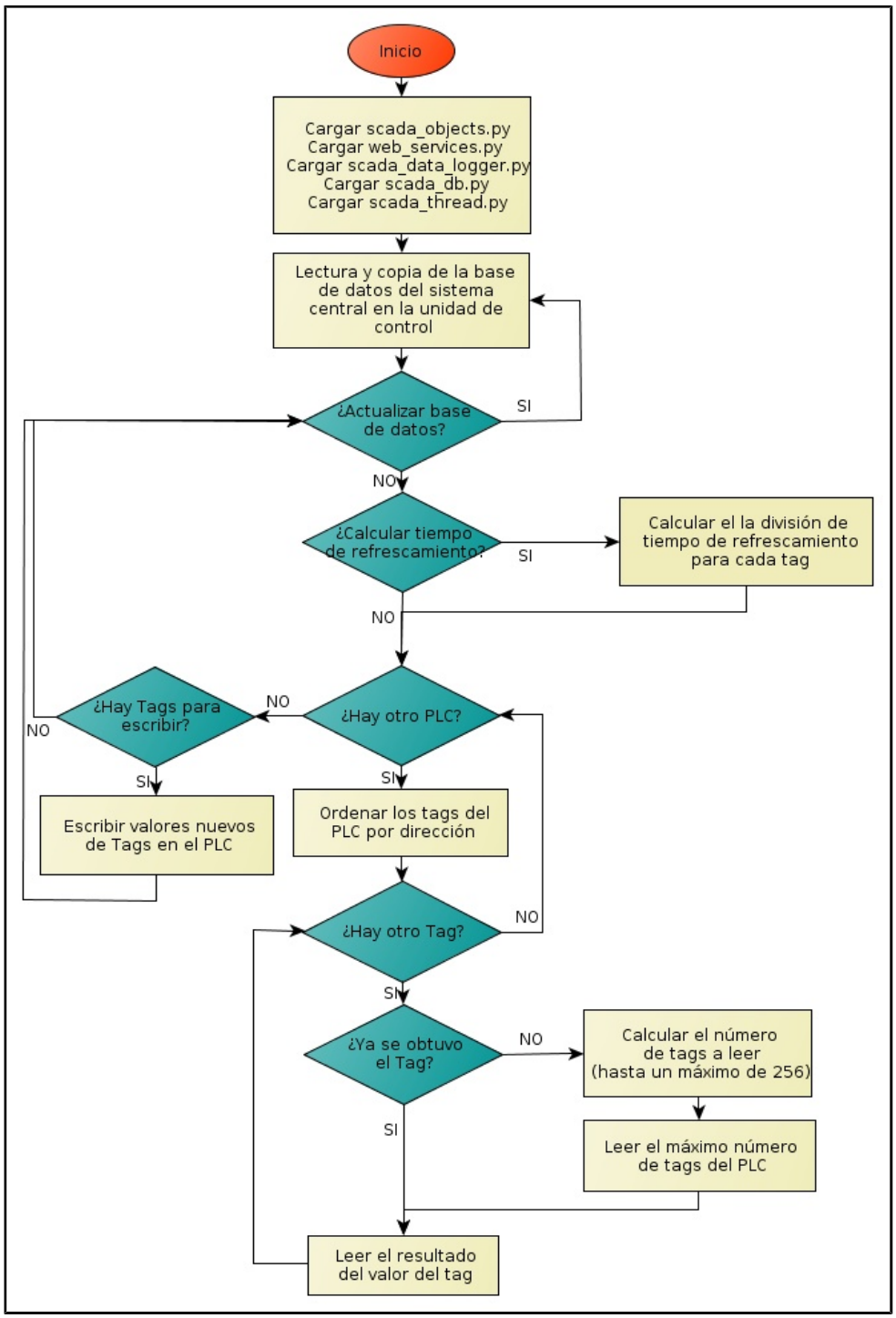


Figura 7. Diagrama de flujo del módulo de la unidad de control.

Para funcionar, el módulo de la unidad de control debe instalarse en el OpenERP. Al estar instalado, este módulo inicia automáticamente con el arranque del programa.

El sistema empieza por cargar todos los archivos y librerías necesarios. Estos controlan el manejo de objetos, el servicio web específicos para el sistema SCADA, el *log* de datos para enviar al sistema central y la base de datos. Luego se busca en la base de datos del sistema central la configuración que corresponde a esta unidad de control específica y se descarga toda la información necesaria para auto configurarse. Si el usuario así lo define se puede recargar la información del sistema central luego durante la ejecución.

El ciclo de trabajo de la unidad de control empieza con la actualización manual de la base de datos. El siguiente paso es el cálculo del tiempo de refrescamiento del sistema, que es el tiempo que debe tardar cada ciclo de trabajo. Este tiempo se calcula utilizando el máximo común divisor entre los tiempos de refrescamiento de cada variable del sistema. El cálculo solo se repite si se cambia alguna variable.

Luego se lleva a cabo la lectura de cada PLC. Para evitar choques de datos en el puerto se lee solo un PLC a la vez y un ciclo controla que se lean todos los PLCs. Para agilizar la lectura de las variables (*tags*) se leen en grupo del PLC y luego, de la lista de resultados se extrae el valor para cada una.

Para evitar errores de lectura se calcula una cantidad máxima de variables para cada llamada al PLC. Solo se pueden leer los tags que estén consecutivos, por lo que en muchos casos se recuperan algunos datos que no se necesitan y luego se descartan. Sin embargo el tiempo de lectura es similar si se lee una sola variable o 256, así que el tiempo de comunicación con el PLC y el tiempo del ciclo de trabajo se reducen considerablemente incluso si hay datos que no se necesitan.

El valor recuperado para cada variable se guarda en los objetos en memoria para tener un control en tiempo real del sistema. También se escribe el dato en el *log* para luego enviarlo al sistema central y que este guarde y maneje los históricos.

Al final de cada ciclo de lectura, cuando ya se han leído todos los PLCs, se analiza la cola de escrituras y se cambia en el PLC el valor para las variables que se modificaran. La razón para no cambiarlas en el momento en que se da la orden sino hasta el final del ciclo es para evitar la perdida de los datos ya leídos de cada tag cuando se lee un grupo y para evitar choques de comunicación que puedan afectar el canal de comunicación y el comportamiento del sistema.

A continuación se explica con mayor detalle cada una de las partes del sistema y se da un enfoque más preciso para cada comportamiento del sistema.

5.4.1. Objetos de la unidad de control

La unidad de control tiene un hardware limitado, por lo que es importante mantener el sistema al mínimo en uso de recursos. Uno de los puntos de mayor consumo es el acceso a la base de datos. Por esta razón se creó una serie de objetos que imitan las relaciones de la base de datos. Estas entidades permiten

tener en memoria los datos sobre las características del sistema que maneja la unidad de control y reducir al mínimo las consultas a la base de datos. El diseño relacional de estos objetos obedece al diseño relacional que usa la base de datos y se muestra en la figura 8.

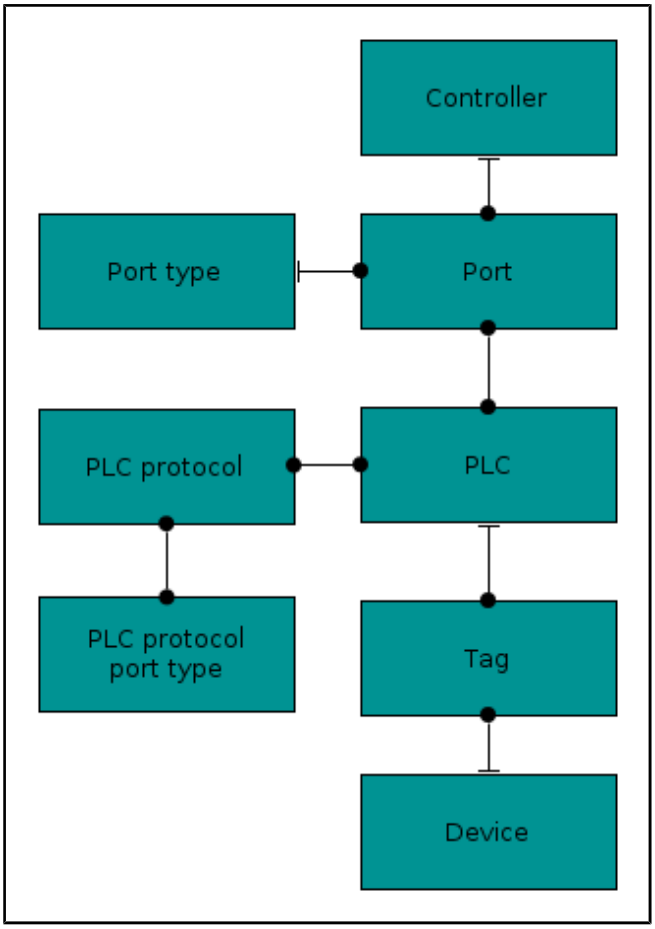


Figura 8. Diseño relacional de objetos de la unidad de control.

Cada objeto corresponde a una parte del sistema y guarda la configuración y atributos de esta parte. El esquema utilizado permite buscar fácilmente los atributos que se necesitan y mantiene un modelo ordenado de la realidad. A continuación se lista cada elemento y se explica la información que se maneja en el objeto.

- **Controller**: Se refiere a la unidad de control. Sus atributos corresponden a información y configuraciones propias de la unidad.
- **Port**: Corresponde al puerto de conexión entre la unidad controladora y los elementos de control.

- **Port type**: Define el tipo de puerto. Se utiliza para acomodar el diseño y para relacionar diferentes puertos.
- **PLC**: Se refiere al elemento de control que se supervisa. En el caso de éste proyecto, todos los elementos que se controlan son PLCs, pero para aplicaciones futuras puede tratarse de otro sistema diferente de control. Tiene información sobre el sistema físico para información.
- **PLC protocol**: Es el protocolo de comunicación que utiliza el elemento de control anterior. Tiene toda la información que el sistema de comunicación necesita para poder realizar una conexión entre la unidad controladora y el elemento que supervisa.
- **PLC protocol port type**: Corresponde al tipo de puerto que puede utilizarse para determinado protocolo. Se utiliza para acomodar el diseño y para relacionar diferentes puertos.
- **Tag**: Se refiere a la variable a supervisar. Contiene información sobre el valor de la variable, su dirección en el PLC y otras informaciones y atributos necesarios para operar y mostrar la variable.
- **Device**: Este objeto corresponde a un dispositivo o sistema real que controlan los PLCs. Es utilizado por el sistema como medio de orden y para buscar y mostrar informaciones al usuario referentes a un solo dispositivo y no a todo el sistema que se está controlando. Contiene información sobre el dispositivo real que pueda ser relevante.

El uso de objetos tiene varias ventajas contra el uso de la base de datos directamente. Como ya se mencionó, el uso de memoria y procesador es menor, pues las llamadas a la base de datos consumen más recursos. Además los objetos pueden tener sus propias funciones y heredar de otros objetos, lo que facilita la manipulación de los datos.

Las entidades creadas guardan su configuración y atributos así como la relación que tienen con los otros objetos, lo que permite buscar y direccionar rápidamente un elemento para conocer algún atributo aunque solo se conozca alguno de los objetos con que se relaciona. Para hacer esto se utiliza direccionamiento por referencia entre las entidades, lo que permite por ejemplo, manipular toda la lista de Tags que tiene un PLC sin necesidad de conocer necesariamente el identificador de cada tag.

Existen una serie de métodos que son heredados de una entidad base llamada *BaseObject*. Si bien algunos objetos modifican y sobrescriben estos métodos para adaptarse a su morfología específica, la mayoría utilizan los básicos y todos cumplen las mismas funciones que se describen a continuación:

- **get**: Recibe el nombre de un atributo del objeto y devuelve el valor de este atributo.

- **set**: Cambia el valor de un atributo dado por el que se le pasa como parámetro. Devuelve el nuevo valor del atributo.
- **delete**: Borra el atributo dado y elimina también su valor de la base de datos. Devuelve “True” si el atributo se pudo borrar.
- **load**: Recibe el nombre de un atributo y busca su valor en la base de datos para actualizarlo. Devuelve “True” si el atributo se actualizó sin problemas.
- **load_all**: Es igual que *load* pero actualiza todos los atributos que se encuentren en la base de datos para ese objeto.
- **save**: Escribe el valor de un atributo en la base de datos.

Este esquema de objetos permite manipular los atributos del sistema completo sin necesidad de estar accediendo a la base de datos, pero manteniendo una posibilidad de utilizarla en caso necesario.

5.4.2. Control de la base de datos

La base de datos está administrada por PostgreSQL y el OpenERP tiene un manejo muy robusto de los datos, que es una de las razones por las que se decidió trabajar con el núcleo de este sistema.

Todos los datos se maneja con objetos y clases de Python. La conexión con la base de datos PostgreSQL y la persistencia de datos se logra con un sistema de mapeo relacional de objetos (ORM: *Object-Relational Mapping*, en inglés).

El ORM permite una mayor mantenibilidad del programar y hace que sea el OpenERP el que se encargue de la base de datos mientras el programa maneja los datos y tablas como si fueran objetos de Python.

El esquema relacional de objetos en la base de datos de la unidad de control es igual al que se muestra en la figura 8, que se utiliza para los objetos en memoria. Esto porque los objetos en memoria tienen una copia de los de la base de datos, pero no se conectan a esta.

Los objetos en la base de datos existen para proporcionar una persistencia de las configuraciones del sistema. La unidad de control los carga al inicio y solo los recarga o escribe si es necesario cambiar algún atributo. Los valores de las variables controladas nunca se escriben en base de datos, pues sus valores son cambiantes y se manejan en el los objetos de memoria para tener datos en tiempo real o en el sistema central para análisis de históricos.

5.4.3. Comunicación con los controladores

La unidad de control debe supervisar diferentes dispositivos controladores. Para el caso particular de este proyecto se trata de un PLC maestro y dos PLCs esclavos controlados a partir del maestro. Todos son de la marca Delta Electronics y están programados para controlar distintos sistemas de una casa de habitación.

El PLC maestro es un modelo DVP-SC [17] con puerto RS-232 para comunicación por protocolo modbus RTU, que es el utilizado para transmisión de datos con la unidad de control. También tiene un puerto RS-485 que también utiliza modbus, por el que se controlan los otros PLCs. Los controladores esclavos son un modelo DVP-ES [18] y otro modelo DVP-SS [19].

Para el manejo de la comunicación por protocolo modbus en Python se utilizó la biblioteca pymodbus en su versión 0.9.0, publicada bajo la licencia BSD [20]. Esta biblioteca define funciones sencillas para hacer llamadas en el protocolo Modbus en cualquiera de sus versiones, ya sea ASCII, RTU o TCP.

Al tratar de leer un dato en determinado registro del PLC con la biblioteca pymodbus se deben cumplir dos pasos:

1. Leer el PLC y obtener una respuesta de éste.
2. Leer el resultado consultado del registro de respuestas.

El primer paso es en el que se construye la llamada por protocolo modbus y se lee la respuesta que devuelve el PLC. Sin embargo esta respuesta se guarda en un registro de respuesta y lo que devuelve esta función es un código de operación. Este código permite saber si hubo un error en la transacción y no se obtuvo resultado. El segundo paso procura obtener el resultado del dato en caso en que la respuesta del primer paso no sea un código de error. Si se trata de leer el resultado cuando hubo un error al llamar al PLC se obtiene un error en el sistema.

Para facilitar el uso de la biblioteca, se programó una clase de Python llamada ModbusClient. Esta clase se construyó para que en la inicialización se creara el cliente modbus y que las funciones internas facilitaran el cumplimiento de los dos pasos de lectura sin errores.

La función "*call*" permite hacer la llamada al PLC y los datos que se obtienen se guardan en el registro de respuestas. La función "*read*"devuelve los valores de los datos que están en el registro de respuestas.

Dado que las funciones para leer bits o registros del PLC son diferentes en la biblioteca, entonces se debe especificar cual función se va a usar en los parámetros de las funciones "*call*" y "*read*" para evitar errores.

5.4.4. Registro del historial de datos

El manejo del historial de los datos puede llegar a contemplar muchos datos muy rápido y depende del sistema que se esté monitorizando. Dado que la unidad de control no tiene tantos recursos para manejar tantos datos y su respectivo procesamiento, este análisis se hace en el sistema central. Ahí se guarda en base de datos todo el historial de eventos y se puede generar gráficos, tablas u otras representaciones de los datos que se desee mostrar.

Como el sistema central no está directamente conectado con los controladores y puede que no esté en todo momento en comunicación con la unidad de control, es necesario que se guarden los datos de alguna manera temporal en la unidad de control y poder enviarlos al servidor en el momento en que se le soliciten.

La forma más ligera y sencilla, que evita la saturación de la memoria RAM es construir un archivo de registros en que se lleve un evento por línea. En este archivo se guarda el dato de tiempo en que ocurrió el evento o que se leyó el dato, el nombre del tag u objeto que se haya modificado, el atributo modificado y el nuevo valor.

Las funciones para el registro de datos se encuentran en una clase de Python llamada *DataLogger*. Esta clase cuenta con cuatro funciones básicas que permiten registrar los datos y enviarlos al servidor central cuando se solicita.

Al enviar los datos se hace un respaldo del archivo de registro y se inicia un nuevo registro de datos. Esto para eliminar los últimos registros y para evitar que se pierdan en caso de problemas de comunicación. Así, si el servidor central no recibe todos los datos, puede solicitar que se le envíen de nuevo.

5.4.5. Servicios web

Los servicios web son herramientas que permiten construir una aplicación distribuida por medio de la red. En estos casos se usa la red como una capa de transporte de datos entre varias partes del sistema. El protocolo XML-RPC es un protocolo muy simple que permite compartir información y procesamiento de datos del sistema entre diferentes equipos. La ventaja que tiene es que puede ser usado en diferentes sistemas operativos y con diferentes lenguajes de programación.

El OpenERP está diseñado como un sistema cliente/servidor, donde los clientes se comunica con el servidor por medio de XML-RPC. De esta forma se pueden hacer llamadas remotas a procedimientos del servidor y este puede responder, todo esto por medio de archivos codificados en XML y transportados por HTTP.

En este caso el cliente es muy sencillo porque solo requiere hacer llamadas a procedimientos en el servidor por medio de XML-RPC y utilizando una serie de parámetros. El servidor es el que se encarga de procesar y analizar los datos y los devuelve al cliente que los muestra al usuario.

El OpenERP ya posee muchas funciones definidas entre los servicios web para acceso al sistema y a la base de datos. También hay un servicio exclusivo para la conexión con los objetos del sistema y sus funciones. Estos servicios ya presentes en el programa se utilizaron para cargar la base de datos del sistema central.

Para la implementación de la unidad de control se creó un nuevo servicio web, como parte del OpenERP, que permite hacer llamadas a procedimientos de

los objetos en memoria y del ciclo de trabajo de la unidad de control. Este sistema brinda funciones que interactúan directamente con los objetos en memoria o que devuelven información de estos objetos para mostrarla al usuario. Las funciones que implementa son:

- **get**: Devuelve el valor de un atributo especifico de un objeto dado.
- **set**: Cambia el valor de un atributo especifico de un objeto dado por el valor especificado.
- **delete**: Elimina un atributo dado de un objeto.
- **get_all**: Devuelve una lista de todos los tags que maneja el sistema, junto con su valor actual. Se utiliza para actualizar los datos de tiempo real.
- **get_attributes**: Devuelve una lista con todos los atributos de un objeto dado.
- **get_device**: Si se especifica un dispositivo (*device*), esta función devuelve la lista de *tags* pertenecientes a ese dispositivo. Si no se especifica entonces devuelve una lista de los dispositivos del sistema.
- **get_objects**: Devuelve una lista con todos los objetos de un tipo dado del sistema.
- **get_data_log**: Devuelve las lineas del registro histórico de datos y reinicia el registro. Siempre se guarda una copia del último registro en caso de que haya un problema en la comunicación.
- **get_data_log_again**: Esta función devuelve las lineas del registro anterior de datos y solo se debe llamar si la función anterior dio algún error.
- **update_db**: Esta función da la orden de recargar la base de datos del sistema central de forma manual.
- **start_thread**: Esta función permite reiniciar el hilo del sistema SCADA manualmente en el caso en que se detenga por un error y no se reinicie automáticamente. Esta función existe para depuración del sistema y no debería ser necesario usarla en sistemas en producción.

5.4.6. Interfaz web de la unidad de control

El sistema SCADA está diseñado para brindar una interfaz al usuario sencilla y completa con acceso tanto a la visualización del sistema como a la configuración. Esta interfaz está albergada en el servidor central y permite ver el estado del sistema en tiempo real y el historial de eventos o de los datos.

Sin embargo pueden existir casos particulares en que no se tenga acceso a esta interfaz. Como cuando se pierde la conexión entre el servidor y la unidad de

control. Para estas situaciones es que se programó una interfaz especial para la unidad de control.

No se utiliza la interfaz web de OpenERP, como en el servidor central porque esta consume mucha memoria RAM si se abren varias sesiones. Además el sistema que se tiene ya montado ocupa cerca de 900 MB en una tarjeta de 1 GB, lo que deja sin espacio para más opciones.

Se utiliza entonces el programa Webmin, que es una interfaz gráfica para el manejo del sistema Linux u otros sistemas operativos por medio de la web. Para poder manejar la unidad de control, se programó un módulo de Webmin que utiliza el protocolo XML-RPC para obtener informaciones o modificar el sistema por medio de los servicios web.

El módulo para la unidad de control está programado en Perl y para las configuraciones se vale de una interfaz de línea de comandos simple que permite conectarse a los servicios web del OpenERP y recuperar las respuestas que da el programa. Esta interfaz de linea de comandos está desarrollada en Python y permite comunicarse con cualquier servicio web del OpenERP, pues todos los parámetros deben definirse en la línea de instrucción para que esta interfaz construya el mensaje XML-RPC. También recibe y decodifica la respuesta y la entrega al usuario como texto.

La interfaz web puede hacer llamadas a los servicios web del OpenERP por medio de esta interfaz e interpreta las respuestas para construir lo que se le muestra al usuario.

El usuario por su parte puede modificar los campos de texto que se le ofrecen para cambiar los diferentes atributos y configuraciones del sistema. Esta interfaz se observa en la figura 9, donde se puede interpretar su funcionamiento por medio de campos de texto y botones.

Para la interfaz de supervisión y control se hace uso de imágenes SVG dinámicas. Estas imágenes permiten utilizar código Javascript para darles funciones que permiten al usuario interactuar con la imagen y el sistema. Estas imágenes son las mismas que integra la interfaz gráfica del sistema central para el control en tiempo real.

Cada imagen SVG debe ser personalizada independientemente para que realice las funciones que se quiera en cada caso particular. Sin embargo, la conexión con la unidad de control para obtención y manipulación de las variables del sistema y otras configuraciones se hacen por medio de los servicios web del OpenERP.

Figura 9. Interfaz de Webmin donde se muestra una de las pantallas de configuración.

La programación de funciones de las imágenes SVG está escrita en lenguaje Javascript y depende de cada imagen independientemente, pues depende de las variables que se necesiten y los objetos que se manipulan en cada imagen. Para la comunicación con los servicios web del OpenERP en la unidad de control se utilizan pequeños scripts programados en PHP que son invocados desde el código Javascript y que funcionan por parámetros, lo que permite que los utilice cualquier nueva imagen SVG que los necesite sin necesidad de modificar el código PHP.

El cliente web del OpenERP no se puede cargar en el sistema de la unidad de control, como se mencionó antes. Sin embargo se puede instalar en otra computadora y configurarlo para que funcione como cliente web de la unidad de control. Este cliente permite un acceso a la base de datos del OpenERP en la unidad de control y con él se puede modificar y controlar los datos del sistema.

Esta opción se puede utilizar en casos en que el sistema presente fallos o no haya conexión a internet. Este cliente puede brindar una interfaz muy similar a la interfaz principal del sistema SCADA, que está junto con el sistema central, lo que daría un ambiente más familiar para el operario. Sin embargo requiere que el cliente web esté instalado y configurado en la máquina con que se va a acceder. Por esta razón se menciona esta opción y se deja la posibilidad de usarla, pero se mantiene como principal la interfaz de Webmin que se discutió anteriormente.

Capítulo 6. Análisis de resultados

En esta sección se analiza el funcionamiento del sistema en una casa de habitación. Esta casa cuenta con un sistema de control de iluminación, control de calentamiento de agua por paneles solares, control de presión constante y control de filtro y calentamiento de la piscina. Existen otros sistemas menores controlados pero que no se agregaron al sistema SCADA para las pruebas.

También se compara el costo que tiene la instalación del sistema actual con un sistema que utilizara IntegraXor, para ver las ventajas competitivas que tiene este nuevo sistema en el mercado.

6.1. Sistema de control de una casa de habitación

La figura 10 muestra las categorías de los diferentes dispositivos de la casa que se supervisan. Se observan cinco diferentes categorías para procurar una monitorización más eficiente y sencilla. Lo primero que se maneja es el sistema de luces de la casa. Este está dividido entre las luces del primero y el segundo piso. Para su control se dibujó un gráfico en formato SVG que representa un panel de botones separados en subcategorías según las diferentes zonas de la casa. Para el funcionamiento de cada interruptor se utiliza una variable digital del PLC y solo se cambia su valor en caso de que se quiera encender o apagar la luz correspondiente.

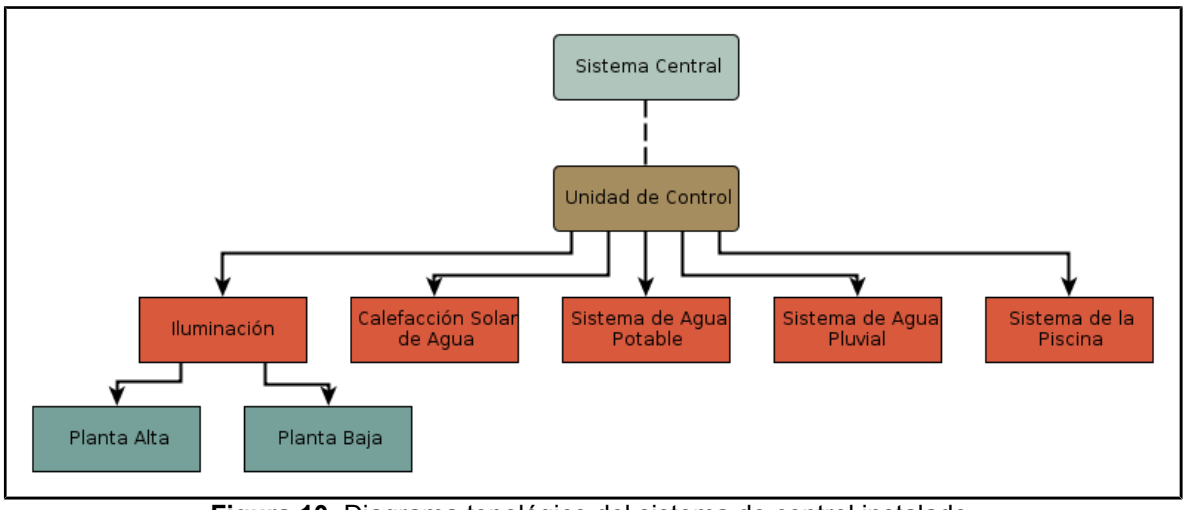


Figura 10. Diagrama topológico del sistema de control instalado.

Luego está el sistema de calentamiento de agua. Este sistema se compone de dos tanques eléctricos de calentamiento y un sistema de recirculación del agua por un panel de calefacción solar. El panel solar es el que se usa como método

principal para calentar el agua y las resistencias eléctricas de los tanques solo se utilizan si no hay energía solar suficiente para calentar. El control de este sistema está programado en el PLC principal pero la unidad de control desarrollada lo puede afectar variando diferentes parámetros de temperatura o encendiendo y apagando los distintos dispositivos.

Se dibujó otra imagen para la supervisión de este sistema donde se muestran las diferentes partes y las temperaturas de este. En la imagen se puede manipular como botones los diferentes dispositivos para encenderlos y apagarlos, como las resistencias eléctricas o las bombas de recirculación de agua. Estos funcionan igual que las luces. Para la manipulación de las variables numéricas del sistema se hace por medio de cuadros de texto que el sistema interpreta como el nuevo valor y lo cambia en el PLC.

El sistema de control de presión constante de agua potable procura mantener una presión de agua en la tubería suficiente todo el tiempo. Lo que hace es utilizar un tanque del que se bombea agua para subir la presión en caso de que sea muy baja. Todo el control está en uno de los PLC esclavos y solo se pueden modificar algunos parámetros básicos del sistema. También existe una imagen para monitorizar este sistema en la que se presenta la presión del agua, el tanque y el nivel de llenado y la válvula solenoide que controla la entrada de agua desde la toma del AyA. Esta válvula se puede cerrar manualmente para forzar el funcionamiento de la bomba del tanque ya sea para vaciarlo y limpiarlo o solo para hacer recircular el agua. El funcionamiento es igual que para las imágenes SVG anteriores.

El sistema de riego y reutilización del agua llovida es muy similar al sistema de agua potable, pero solo se usa para situaciones específicas como el riego o las descargas de los inodoros. El sistema controla cuanta agua hay en el tanque propio y la utiliza mientras quede suficiente. En caso de agotarse el sistema utiliza el agua potable en las tareas necesarias. La imagen donde se supervisa este sistema muestra la presión y el nivel del tanque de agua llovida y muestra si está en funcionamiento la bomba del tanque. También permite controlar si el sistema de riego está funcionando y ponerlo a funcionar en caso de que se requiera.

El último de los sistemas que se supervisa es la piscina. Esta cuenta con un filtro y una luz, pero además cuenta con un sistema de calefacción del agua por medio de energía solar. La supervisión y manipulación de este sistema se puede hacer desde la imagen SVG preparada para éste, donde se muestra un diagrama simplificado de su funcionamiento con las diferentes partes a monitorizar.

La unidad de control es la que maneja y refresca todas estas variables constantemente. Por otro lado, las imágenes SVG, y los *scripts* de Javascript asociados solo piden los datos para mostrarlos. En caso de ordenar una modificación desde la interfaz que crean estas imágenes, La unidad de control espera a terminar las lecturas y luego revisa la cola de escrituras que exista para hacer las llamadas al PLC que corresponda para escribir en estas variables. Este

funcionamiento se muestra en el diagrama de flujo de la figura 7, que se describió en el capítulo anterior.

Tomando en cuenta todos los sistemas que se están monitorizando, se tiene un total de 188 tags a controlar desde los tres PLC del sistema. La lectura de los tags no se hace uno por uno, como se explicó en el capítulo anterior, sino que se leen grupos de tags del PLC para disminuir el tiempo de lectura. Sin embargo, cuando los tags están en lugares muy alejados dentro de la lista de variables del PLC, la cantidad de lecturas aumenta, lo que hace que el sistema se vuelva más lento para refrescar todos los valores.

En el caso particular, las variables que tiene el sistema no están todas adyacentes, sino que están repartidas y en algunos casos muy separadas entre ellas. Esto causa que el sistema realice 12 lecturas al PLC. Cada lectura tarda en promedio 70 ms en realizarse, por lo que el tiempo que se tarda para refrescar todos los los tags es en promedio 840 ms. Dado esto, lo que se tarda para que el sistema responda y refresque por una orden dada en la interfaz de usuario no puede ser menor a 500 ms. En realidad el ciclo de trabajo total, que corresponde a la lectura de los tags y escritura, en caso necesario, tarda entre 800 ms y 950 ms y se puede ver en forma de advertencia en el log del sistema, que lo considera un error pues se pretendía que fuera menor a 500 ms.

Sin embargo, el experto de la empresa SIESA que asesoró el proyecto considera que incluso un tiempo de refrescamiento de un segundo sigue siendo un buen tiempo para un sistema SCADA y está satisfecho con el resultado. Por esta razón se considera que obtener hasta 950 ms es un buen resultado incluso cuando no se cumplió el objetivo.

Nótese que existe una forma de mejorar este tiempo, pero implica una reorganización de todas las variables en el PLC, lo que sería un trabajo tedioso. La forma de hacerlo es colocar las variables que se utilizan lo más cercanas posible para que se tenga que leer el mínimo de veces el PLC al refrescar.

En el caso de la casa de habitación, por ser solo 188 variables, se podrían acomodar de tal forma que solo haya que leer el PLC dos veces, la primera leería todos los registros de un bit y la segunda leería todos los registros numéricos, dado que estas lecturas tienen que hacerse separadas y se puede leer hasta 256 variables adyacentes. Esto reduciría el tiempo de lectura considerablemente a solo 140 ms y posiblemente el tiempo del ciclo de trabajo a un máximo de 250 ms. Estos últimos datos no corresponden a ninguna medición y se obtuvieron suponiendo que se mantienen condiciones similares en el sistema.

De hecho, si se toman las 12 lecturas que se hacen al PLC, y dado que en las pruebas se constató que el tiempo de lectura es similar y en promedio 70 ms sin importar si se leen 1 o 256 variables adyacentes, se puede suponer también que se podrían leer hasta 3072 tags en un ciclo de una duración similar a la que se tiene en el sistema actual.

En esta implementación en realidad no se leen las 3072 variables que se podría leer, pero si se leen muchas más que 188, pues, como se explicó en el capítulo anterior, es más eficiente leer grupos de variables adyacentes en el PLC y descartar los valores de las variables que no se usan, que leer solo los grupos adyacentes que si se usan o una sola a la vez.

6.2. Comunicación del sistema con el servidor central en la nube

En el capítulo cinco se mencionó la forma en que la unidad de control maneja el historial de datos para las variables en forma de un archivo de texto con una entrada con fecha por linea. El servidor central por su lado es el que se encarga de administrar los datos históricos de todos los tags. Este sistema central se encuentra instalado en un servidor en internet y por medio de un servicio web solicita los datos históricos y actuales de los tags del sistema cada minuto.

Originalmente se quería lograr obtener los datos históricos de la unidad de control cada segundo, como se indica en los objetivos. Se trataba de tener un servidor central actualizado al estado más cercano a la realidad posible. Sin embargo, al definir este objetivo específico no se conocía bien el sistema OpenERP, el cual permite crear tareas que se ejecutan cada cierto tiempo, pero el tiempo mínimo que permite es de un minuto.

Para resolver el problema de control y supervisión de datos en tiempo real, que surge de tener el servidor retrasado hasta un minuto del estado del sistema, se decidió que el servidor central manejara los historiales, que el sistema empotrado, por su limitada memoria no es capaz de manejar, pero que el control en tiempo real se colocara en la unidad de control.

El control y supervisión en tiempo real corresponde a la interfaz gráfica por medio de imágenes en formato SVG interactivas con código en Javascript y código PHP para la comunicación con los servicios web del sistema.

La ubicación de la interfaz de control y supervisión en tiempo real en la unidad de control resolvió también otros problemas que tenía el sistema que no se habían contemplado en el diseño inicial, como el retraso de la interfaz de control y supervisión por la latencia de la red o el hecho de tener que utilizar una interfaz muy distinta a la del servidor central en la red local cuando no se tiene conexión a internet o al servidor.

Por estas razones, se considera que aunque el objetivo de mantener el servidor central actualizado cada segundo no se cumple, se logra igual obtener con un sistema de supervisión y control en tiempo real y se solucionan otros problemas que hubieran surgido incluso teniendo el sistema en la nube actualizado cada segundo.

6.3. Consumo de energía de la unidad de control

El consumo de energía de la unidad de control se midió utilizando un medidor de energía *VIP Energy* de la marca *ELCONTROL [21]*. Este medidor es bastante completo y permite medir la tensión, corriente, potencias, factor de potencia y consumo energético de un sistema monofásico o trifásico. En este caso solo se tomaron datos de consumo de energía del sistema.

También se midió el consumo de una computadora de escritorio en la que estaba instalado el sistema IntegraXor que se usaba anteriormente en la casa de habitación.

A continuación se resumen los datos que se obtuvieron para el consumo de energía de cada dispositivo en estado inactivo y con el sistema de cada uno trabajando.

Tabla 2. Consumo de energía de una computadora personal y de la placa de trabajo ALIX.

Energía consumida	En inactividad (W)	Mientras trabaja (W)
Computadora personal	105	128
ALIX 6E1	6,2	12,6

Se puede ver fácilmente en la tabla anterior que la placa empotrada ALIX es un sistema de muy bajo consumo de energía, su consumo es más de 10 veces menor que el de la computadora personal mientras trabaja. Y casi 17 veces menor en estado inactivo.

Se decidió inicialmente trabajar en un sistema empotrado de bajo consumo de energía para evitar el gasto innecesario de energía que provoca tener una computadora de escritorio encendida las 24 horas del día supervisando el sistema. Se puede notar, en la tabla 2, que el sistema que se implementó tiene un consumo mucho menor y por lo tanto representa un ahorro considerable en la factura energética.

6.4. Comparación de costo de un mismo proyecto utilizando el sistema SCADA creado o el sistema IntegraXor

El anexo A plantea los costos que tiene un proyecto en el cual se instala el sistema SCADA a una casa de habitación. Particularmente menciona dos casos, utilizando el sistema desarrollado o el sistema IntegraXor. Los costos de estos proyectos se muestran en la tabla a continuación.

Tabla 3. Resumen de costos de un proyecto de implementación de un sistema SCADA.

Tipo de proyecto	Costo ($)
Proyecto con el sistema SCADA desarrollado	6947
Proyecto con el sistema IntegraXor	7517

La diferencia de costos entre los dos proyectos es de $ 570, esta diferencia hace notar que el sistema SCADA desarrollado es menos costoso en cuanto a la inversión inicial. También se puede analizar el costo de mantenimiento, el cual, como se puede leer en el anexo A, es igual para los dos sistemas. Por esta razón se puede concluir que el sistema desarrollado representa una ventaja competitiva económicamente contra otros sistemas, en este caso IntegraXor.

Debe notarse también que todos estos costos se calcularon utilizando la licencia del sistema IntegraXor para 256 tags y 2 usuarios solamente. Si el sistema requiere más usuarios o controlar más variables entonces el precio de la licencia de IntegraXor subiría. Para el sistema desarrollado no existen estas limitaciones y se puede trabajar con cuantos tags o usuarios se requiera. En el caso de expandirse hacia sistemas más grandes, la diferencia de costos entre las dos soluciones para el proyecto se agrandaría.

Capítulo 7. Conclusiones y recomendaciones

7.1. Conclusiones

- El diseño de la unidad de control como un modulo de OpenERP facilita el manejo de bases de datos y simplifica la programación y posible modificación por parte de cualquier desarrollador que conozca el funcionamiento básico del OpenERP.
- La comunicación con los PLC demuestra que se puede trabajar de forma muy eficiente para su lectura al leer grupos de variables en vez de leer una por una. Sin embargo se observan las limitaciones que tiene el protocolo Modbus que solo permite la lectura de hasta 256 variables por transacción, lo que puede afectar el tiempo que se tarda en refrescar el valor de todos tags del sistema cuando no se ordenan de forma que se propicie el menor número de lecturas.
- Los servicios web y el protocolo XML-RPC representan una forma muy sencilla y rápida de comunicar diferentes sistemas que deben trabajar en conjunto o compartir información por medio de una red local o Internet. La facilidad para crear nuevos servicios web en el OpenERP representa una ventaja al desarrollar nuevos módulos cuya funcionalidad difiere del funcionamiento normal de un sistema ERP.
- El tiempo mínimo de un minuto para la ejecución de trabajos programados en el OpenERP es una limitante para mantener el servidor central actualizado. Sin embargo se pudo resolver con la creación de registros de históricos temporales en la unidad de control. Esto no es lo ideal, pues el sistema central llega a tener hasta un minuto de retraso en el historial del sistema, pero presenta una solución que además evita que hayan perdida de datos en caso de una interrupción prolongada en la comunicación entre el servidor central y la unidad de control.
- El consumo de energía de una placa empotrada ALIX 6E1 es sumamente bajo, apenas 12.6 W, lo que permite obtener un sistema muy económico en cuanto al consumo de energético.
- El costo de instalación del sistema SCADA desarrollado es menor que $7500 que es el costo para un sistema que utiliza IntegraXor, esto le da una oportunidad competitiva al nuevo sistema en el momento en que salga al mercado.

7.2. Recomendaciones

- Un desarrollo del módulo del sistema SCADA en un eje más administrativo permitiría que se pueda integrar en un sistema completo de manejo de recursos empresariales y se pueda acoplar el control y supervisión industrial al manejo y planificación de dichos recursos.
- Una reorganización de las variables en la memoria del PLC permitiría minimizar la cantidad de lecturas que se hacen de este y así se podría reducir considerablemente el tiempo que se tarda refrescando el valor de todos los tags del sistema.
- Se recomienda realizar pruebas de estrés al sistema para documentar el funcionamiento en casos críticos de trabajo y definir su umbral de seguridad.

Bibliografía

[1]. Daneels, A.; Salter, W.. 1999. ***What is SCADA?***. Informe presentado a International Conference on Accelerator and Large Experimental Physics Control Systems. CERN. Trieste, Italia. URL: https://accelconf.web.cern.ch/accelconf/ica99/papers/mc1i01.pdf. Consultado el: 07/07/2011.

[2]. Health, S. 2003. ***Embedded systems design***. Segunda ed. Newnes. Oxford, Boston. 430 p.

[3]. The Modbus Organization. 2006. ***MODBUS over serial line specification and implementation guide V1.02***. URL: http://modbus.org/docs/Modbus_over_serial_line_V1_. Consultado el: 07/07/2011.

[4]. Date, C J. 2001. ***Introducción a los sistemas de bases de datos***. Septima ed. Pearson Educación. Mexico. 936 p.

[5]. Gálvez, S. 2011. ***Tipos de Bases de Datos***. URL: http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf. Consultado el: 07/07/2011.

[6]. ECAVA. 2012. ***Integraxor***. URL: http://www.integraxor.com/. Consultado el: 16/01/2012.

[7]. Indusoft. 2012. ***Indusoft***. URL: http://www.indusoft.com/. Consultado el: 16/01/2012.

[8]. Novar. 2012. ***Opus***. URL: http://www.novar.com/ems-bas/opus-building-automation-system. Consultado el: 16/01/2012.

[9]. Likindoy. 2012. ***Likindoy The Free SCADA***. URL: http://www.likindoy.org/. Consultado el: 16/01/2012.

[10]. FreeSCADA. 2012. ***FreeSCADA***. URL: http://www.free-scada.org/. Consultado el: 16/01/2012.

[11]. Centro de Innovación Tecnológica del Aluminio de Venezuela. 2012. ***SCADA ARGOS***. URL: http://www.cintal.com.ve/argos/. Consultado el: 16/01/2012.

[12]. Roman Savochenko. 2012. ***OpenSCADA***. URL: http://oscada.org/. Consultado el: 16/01/2012.

[13]. Voyage. 2011. ***Voyage Linux***. URL: http://linux.voyage.hk/. Consultado el: 07/07/2011.

[14]. Python Software Foundation. 2011. ***Python Programming Language***. URL: http://python.org/. Consultado el: 07/07/2011.

[15]. OpenERP. 2011. ***Technical Architecture***. URL: http://doc.openerp.com/v6.0/developer/. Consultado el: 07/07/2011.

[16]. PC Engines. 2010. ***Alix6e1 product file***. URL: http://pcengines.ch/alix6e1.htm. Consultado el: 18/06/2011.

[17]. Delta Electronics, Inc.. 2011. ***DVP-SC Series***. URL: http://www.delta.com.tw/product/em/control/plc/con. Consultado el: 07/07/2011.

[18]. Delta Electronics, Inc.. 2011. ***DVP-ES Series***. URL: http://www.delta.com.tw/product/em/control/plc/con. Consultado el: 07/07/2011.

[19]. Delta Electronics, Inc.. 2011. ***DVP-SS Series***. URL: http://www.delta.com.tw/product/em/control/plc/con. Consultado el: 07/07/2011.

[20]. Pymodbus Team. 2011. ***Pymodbus: A Modbus Protocol Stack in Python***. URL: http://code.google.com/p/pymodbus/. Consultado el: 07/07/2011.

[21]. Elcontrol. 2011. ***VIP Energy: Manual de usuario***. URL: http://www.elcontrol-energy.net/download.asp. Consultado el: 07/07/2011.

Apéndice A. Estudio económico

El presente anexo tiene como finalidad el calculo acertado del precio del software para el sistema SCADA, Se incluyen los costos tanto de la unidad de control como del servidor central para obtener el precio de un sistema equivalente a cualquier sistema actual en el mercado.

También se calcula el costo de una implementación como la que se utilizó para pruebas del sistema utilizando el software IntegraXor y se compara con el costo de la misma implementación utilizando el sistema del proyecto SCADA con los precios que se calcularon anteriormente.

A.1. Calculo del precio base para el sistema SCADA

Para calcular acertadamente el precio que debe tener el software del proyecto SCADA se deben considerar dos aspectos, el costo del desarrollo inicial del proyecto y el costo para mantener y actualizar el sistema.

En le primer rubro, se toma en cuenta los gastos en que se incurrió para el desarrollo del proyecto, los salarios que se pagaron para los dos practicantes y los costos operativos que significó el proyecto para la empresa ClearCorp. Todos los artículos que se compraron serán reutilizados dentro de la empresa para otros proyectos futuros, por lo que el costo para el proyecto corresponde solamente a la depreciación de estos. Una tabla resumen de los gastos se cita a continuación.

Tabla 4. Gastos en que se incurrió en el desarrollo del proyecto.

Articulo	Precio ($)	Vida útil (años)	Depreciación ($)
Tarjeta empotrada ALIX6E1	195	10	19,5
Tarjeta de expansión de puertos	60	10	6
Otros cables y conectores	50	10	5
PLCs Delta	4567	10	456,7
USB-RS485	175	10	17,5
Servidor en la nube	35	1	35
		Total	539,7

El precio de los PLCs Delta incluye los tres PLCs principales que se utilizaron en la casa de habitación y 25 expansiones de puertos para estos PLCs. Además se incluye un PLC más que se utilizó para desarrollo en la oficina.

Los salarios de los dos practicantes suman el gasto más importante que se dio en desarrollo y se justifican a continuación. Debe notarse que uno de los practicantes trabajó 6 meses mientras que el otro trabajó siete meses.

Tabla 5. Costo de los salario de los dos practicantes durante el desarrollo del proyecto.

Empleado	Salario ($/mes)	Meses trabajados	Total ($)
Practicante 1	500	7	3500
Practicante 2	500	6	3000
		Total	6500

El costo operativo mensual de ClearCorp por cada trabajador es de:

$ 166,98

Con lo que se calcula un costo operativo del proyecto de:

$ 2170,74

Sumando estos tres rubros se tiene un costo de desarrollo del proyecto de:

$$\begin{array}{rr} & 539{,}7 \\ + & 6500 \\ + & 2170{,}7 \\ \hline \$ & 9210{,}4 \end{array}$$

Luego se calcula el costo por el mantenimiento y mejoramiento del sistema, este costo se calcula basado en el salario de un ingeniero que haga este trabajo, el costo operativo de la empresa ClearCorp y la utilidad que se pretende ganar mensualmente. Este costo se ve desglosado en la siguiente tabla.

Tabla 6. Costo de mantenibilidad y actualización del sistema SCADA en $ por mes.

Salario del ingeniero	1500
Costo operativo	166,98
Utilidad	2500
Total	4166,98

La utilidad de $ 2500 se decidió arbitrariamente tomando en cuenta varios parámetros. Primero se desea recuperar la inversión que se hizo en el desarrollo del proyecto en un máximo de cuatro meses, además se pretende un precio cercano a $ 500, pues elevarlo por arriba de esto haría que el producto deje de ser competitivo. Para recuperar los $ 9210,44 que costó el desarrollo en cuatro meses se requiere al menos tener una utilidad de $ 2302,61, lo que fija el mínimo de utilidad que se pretende conseguir.

Para el cálculo de los precios se toma en cuenta que se venderán 100 proyectos al año, dato que se toma del criterio de experto pedido a los gerentes de la empresa SIESA.

Si se utiliza una utilidad mínima de $ 2302,61, el precio del producto para que cubra los gastos mensuales y esta utilidad debería ser de $ 476,35. Este

precio es todavía menor al máximo que se considera aceptable, por lo que permite elevar la utilidad al valor de $ 2500. Para esta utilidad y con todos los demás valores invariantes se calcula un precio del producto de $ 500,04, que se redondea para facilidades de cobro a $ 500.

Así se fija el valor del software del sistema SCADA en 500 $.

A.2. Costo de un mismo proyecto utilizando el sistema SCADA creado o el sistema IntegraXor

El sistema IntegraXor es una solución comercial para sistemas SCADA similar a la que se desarrolló en el proyecto SCADA. Existen muchos otros sistemas de este tipo y se escogió IntegraXor para hacer la comparación porque es el que utiliza SIESA en la casa de habitación en que se probó el sistema.

Se pretende comparar el costo de un proyecto de instalación y mantenimiento de un sistema SCADA con la solución diseñada y con IntegraXor. Se supone para ello un proyecto en el que se instala el control y el sistema SCADA a una casa de habitación similar a la que se utilizó para probar el proyecto. En este caso se tienen 200 tags a controlar.

La licencia del sistema IntegraXor tiene un precio de $ 400 anuales para un sistema de hasta 256 tags y con únicamente dos usuarios. Lo primero que se puede observar es que en el caso del sistema que se desarrolló, el precio calculado es mayor, $ 500. Sin embargo esto no justifica que un proyecto completo igual pueda salir más caro.

Para el sistema SCADA desarrollado se utiliza una tarjeta empotrada ALIX 6E1 y un servidor en la nube. Se puede ver el costo que tendría este proyecto en la tabla siguiente.

Tabla 7. Costo del proyecto supuesto utilizando el sistema SCADA desarrollado.

	Precio ($)	Cantidad	Total unitario ($)
Tarjeta empotrada ALIX 6E1	195	1	195
Tarjeta de expansión de puertos	60	1	60
Otros cables y conectores	50	1	50
PLCs Delta	4567	1	4567
Conector USB-RS485	175	1	175
Servidor en la nube	400	1	400
Software SCADA	500	1	500
Costo de Instalación	1000	1	1000
		Total	6947

Si se utiliza el sistema IntegraXor, debe utilizarse una computadora con ciertas características mínimas. Según el criterio de experto de los ingenieros de SIESA, la computadora debe ser de una gama alta de hardware y su precio está arriba de $ 1500. Se muestran entonces los costos con este sistema.

Tabla 8. Costo del proyecto supuesto utilizando el sistema IntegraXor.

	Precio ($)	Cantidad	Total unitario ($)
Computadora, accesorios y licencias de software	1500	1	1500
Otros cables y conectores	50	1	50
PLCs Delta	4567	1	4567
Licencia de IntegraXor (256 tags, 2 usuarios)	400	1	400
Costo de Instalación	1000	1	1000
		Total	7517

La solución con el sistema IntegraXor cuesta $ 570 más que la misma solución utilizando el sistema desarrollado.

Debe analizarse también el costo de mantenimiento de estos dos sistemas. Se supone que el soporte para cualquiera de las dos soluciones tiene un costo igual. Para la implementación desarrollada lo único que tiene un costo anual es el servidor en la nube y su costo es de $ 400. Para la solución que utiliza IntegraXor las licencias son las que representan un costo anual de $ 400. Comparando el costo anual, se puede asumir que los dos sistemas tienen el mismo costo.

Un punto importante a mencionar es que se utilizó el precio de la licencia del sistema IntegraXor limitada a 256 tags y a dos usuarios. En caso de que se necesitaran más tags o más usuarios el costo de esta licencia sube. En este caso la diferencia de costos entre los dos proyectos sería aun mayor.