Universidad de las Ciencias Informáticas.
Facultad 5

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Título: Herramienta de configuración para el PLC-HMI.

Autora: Lauren San Juan Guía

Tutor: Ing. Roberto Alejandro Espí Muñoz

Co-tutor: Ing. Daisy Diana Vargas Vento

La Habana, 2012

“El único autógrafo digno de un hombre es el que deja escrito con sus obras”.

José Martí.

Declaración de Autoría

Declara ser autora del presente trabajo de diploma y reconocer a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Lauren San Juan Guía

Autora

_____ _____

Ing. Roberto Alejandro Espí Muñoz

Tutor

Daisy Diana Vargas Vento

Co-tutor

Datos de contacto

Tutor:

Nombre y apellidos: Roberto Alejandro Espí Muñoz.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: raespi@uci.cu

Co-tutor:

Nombre y apellidos: Daisy Diana Vargas Vento.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: ddvargas@uci.cu

Autora:

Nombre y apellidos: Lauren San Juan Guía.

Institución: Universidad de las Ciencias Informáticas (UCI).

e-mail: lsanjuan@estudiantes.uci.cu

Agradecimientos

Agradecer en primer lugar a mi familia que durante toda mi vida han apoyado mis estudios y han contribuido en mi formación profesional, personal y en el logro de mis sueños.

A mi madre que siempre sostiene mi mano en buenos y malos momentos. Tu infinito amor y dedicación me devuelve la confianza y me salva cada día.

A mi padre gracias por los sacrificios y por tu apoyo en cada etapa de mi vida y en todos mis sueños. Laura gracias por tu sonrisa, por ser tan buena y tener tanta confianza en mí, espero que veas en tu hermana un ejemplo que te impulse a ser cada día mejor. Gracias a Dios por estar a mi lado a pesar de mis debilidades.

Agradecer a mi tutor Roberto por tener tanta paciencia y apoyarme hasta el final, cuando creí que no sería posible. A Daysi por apoyarme y ayudarme.

A mis abuelos que los quiero mucho, mi tía Rebeca que me ayuda y me anima a no bajar la guardia, a mis tíos todos, a mis primos que me apoyan Vladito, Claudia, Rosmary, Magela, Rachel y Edumis (como si lo fuera) los quiero mucho.

A mi novio que a pesar del poco tiempo, ocupas un lugar especial en mi vida y me has apoyado mucho en el logro de este sueño.

A mis amigos de siempre tienen un lugar muy especial en mi corazón Massiel, Diana, Mayvis. A los nuevos amigos que juntos somos como una gran familia universitaria.

A mis profesores, Millet, Villar, Varcacel, Danilo y muchos otros que han depositado confianza en mí y han contribuido en mi desarrollo profesional.

A todos aquellos que me ayudaron Karel, Leiser, Ybrain, Ridel, a la línea Sistemas Empotrados.

A mis compañeros del teatro y profesores del centro cultural, que juntos hemos pasado momentos muy gratos.

A todos aquellos que de una forma u otra han contribuido al logro de este sueño.

Lauren

Dedicatoria

Dedico esta tesis a mi madre por ser mi guía y mi gran inspiración. A mi papá y mi hermanita que tanto quiero y siempre están conmigo. A toda mi familia, que siempre me apoyan y que están muy orgullosos de mí.

Resumen

En la industria moderna el control de los procesos de forma automática se hace cada vez más necesario para garantizar la eficiencia y la calidad de la producción. Actualmente muchas de las industrias cubanas no poseen un sistema automático que controle sus procesos industriales.

La línea Sistemas Empotrados, perteneciente al Centro de Informática Industrial (CEDIN) de la Facultad 5, propone como solución a este problema que presentan las industrias cubanas, la creación de un dispositivo automático de control y supervisión denominado PLC-HMI, el cual brindará una variante de automatización integral.

Actualmente realizar la configuración que permita fijar los parámetros de ejecución del PLC-HMI para poder crear el entorno apropiado en el que debe operar, constituye una tarea engorrosa. Surge entonces la necesidad de crear un mecanismo que cumpla con las características técnicas del PLC-HMI y facilite la tarea de configurar los parámetros de ejecución para que el dispositivo pueda ser utilizado en el ambiente industrial deseado.

Como resultado del presente trabajo, se realizó el diseño e implementación de una herramienta informática que permita la configuración de los elementos de hardware del PLC-HMI. A dicha herramienta de configuración se le aplicaron diferentes pruebas de software permitiendo detectar y corregir la máxima cantidad de errores antes de su entrega al cliente.

Palabras clave: dispositivo automático de control y supervisión, herramienta de configuración, parámetros de ejecución.

Abstract

In modern industry automatic process control becomes a necessity to guarantee the efficiency and quality of production. Many of the Cuban industries lack an automated control process.

The Embedded Systems development area in the Industrial Informatics Centre (CEDIN) at School 5 of the University of Information Sciences presents as a solution to this problem the creation of an automated device for supervision and control called PLC-HMI with the main objective of delivering an integral solution in automation. It's configuration (mainly the setting of execution parameters) represents a cumbersome task. As a result of this report, the design and implementation of a software application that can configure such a device was achieved. Different test parameters were applied to the application in order to correct as many problems as possible.

Keywords: configuration tool, execution parameters, supervision and control automated device.

Índice

Índice de Figuras

Índice de Tablas

Introducción

En los últimos años, el crecimiento de las industrias ha conllevado a la extensión de las fábricas y maquinarias que las componen, lo que ha traído como consecuencia un aumento en la complejidad de los procesos industriales. La mayoría de estos procesos requieren un control automático para lograr que sus sistemas tengan un mejor desempeño y una mayor eficiencia.

Actualmente muchas de las industrias cubanas no poseen un sistema automático para la operación y el control de la producción. Por otro lado, existen muchas industrias que no requieren procesos complejos a automatizar o no cuentan con elevados recursos económicos para adquirir equipos que le permitan controlar y supervisar sus procesos industriales.

En la Universidad de las Ciencias Informáticas, la producción se encuentra estructurada por centros que integran la formación y la producción en torno a una temática para convertirla en productos de software. En la Facultad 5 se encuentra el Centro de Informática Industrial (CEDIN) constituido por varias líneas productivas.

La línea Sistemas Empotrados de dicho centro, propone como solución a estos problemas que presentan las industrias cubanas, la creación de un dispositivo automático de control y supervisión denominado PLC-HMI como variante de automatización integral. Un PLC-HMI es un Controlador Lógico Programable (PLC) que dispone de una tarjeta basada en microcontroladores denominada CID 300/9, desarrollada por el Instituto Central de Investigaciones Digitales (ICID). El dispositivo es diseñado para controlar procesos secuenciales en tiempo real, además presenta un módulo Interfaz Hombre-Máquina (HMI) que permite la visualización e interacción a los operadores con los procesos automatizados.

Actualmente realizar la configuración que permita fijar los parámetros de ejecución del PLC-HMI, para poder crear el entorno apropiado en el que debe operar, constituye una tarea engorrosa. Esta tarea debe realizarse empleando un lenguaje de bajo nivel y en caso de cambios en el proceso industrial o nuevas prestaciones de hardware, se debe realizar nuevamente la configuración de manera trabajosa. El método de configuración actual ocasiona pérdida de tiempo, empleo excesivo de fuerza de trabajo y un nivel de

conocimiento en programación que no tienen porque dominar los operadores del dispositivo.

Es por esta razón que el presente Trabajo de Diploma se plantea como **problema científico** ¿Cómo facilitar el proceso de configuración del PLC-HMI basado en microcontroladores?

Por lo que esta investigación tiene como **objeto de estudio** las herramientas de configuración y parámetros de ejecución de PLC.

Proponiendo como **objetivo general** de la investigación: Diseñar e implementar una herramienta de configuración del PLC-HMI que permita la comunicación con los dispositivos de hardware.

El **campo de acción** lo constituyen las herramientas de configuración para dispositivos programables basados en microcontroladores.

Conforme a este planteamiento se derivan las siguientes **tareas a desarrollar**:

1. Análisis de las características y funcionalidades de los PLC basados en microcontroladores para obtener una base teórica de estos dispositivos dentro de un sistema de control y supervisión.
2. Definición de los elementos de configuración necesarios de la herramienta a construir a partir del estudio de herramientas similares.
3. Definición de los lenguajes de programación, así como las tecnologías y las metodologías de software necesarias para el diseño de la herramienta de configuración.
4. Diseño de la herramienta de configuración del PLC-HMI para obtener un modelo que describa los aspectos necesarios de la herramienta a construir.
5. Implementación de la herramienta de configuración del PLC-HMI.
6. Realización de pruebas al sistema para evaluar si los resultados son los esperados en el ambiente que está concebido.

Los **Métodos de investigación científica** empleados son:

Métodos teóricos:

- ***Análisis histórico lógico:*** Con el objetivo de analizar las herramientas de configuración de PLC, su evolución cronológica, sus funcionalidades y principales características.
- ***Análisis y síntesis:*** Para realizar el análisis y estudio de las bibliografías existentes sobre el tema en cuestión y lograr obtener de manera sintetizada el contenido necesario y suficiente para la realización del presente trabajo.
- ***Modelación***: Para crear las representaciones que ofrezcan una visión simplificada de la realidad, facilitando la compresión de la propuesta y los elementos del diseño de la solución.

Métodos empíricos:

- ***Consulta de información en todo tipo de fuente:*** Para la elaboración del marco teórico de la investigación.
- ***Realización de pruebas:*** Para evaluar los resultados.

Como **idea a defender** se puede enunciar: La creación de la herramienta de configuración para el PLC-HMI propuesta permitirá configurar de una manera más efectiva los parámetros de ejecución del mismo.

Al concluir este trabajo se espera obtener como **principales resultados**:

- Diseño e implementación de una herramienta para la configuración de los elementos de hardware del PLC-HMI.
- Configuración efectiva de los parámetros de ejecución del PLC-HMI.

El presente documento está estructurado en tres capítulos, a continuación se describe el contenido que se abordará en cada uno de ellos:

Capítulo 1: Fundamentación teórica

EL marco teórico de esta investigación se enmarca en las características de los PLC, su funcionamiento, principales componentes de hardware y sus elementos configurables. También aborda algunos aspectos a tener en cuenta para la configuración de autómatas, así como ejemplos de aplicaciones creadas por fabricantes de PLC que realizan estas

tareas. Otro aspecto importante tratado en esta sección es la definición de las tecnologías y herramientas que se emplean durante el desarrollo del trabajo.

Capítulo 2: Propuesta del sistema. Diseño e implementación

El análisis y el diseño son etapas importantes en el ciclo de desarrollo de software, toda aplicación debe estar sustentada en un adecuado análisis y una correcta selección de su arquitectura. El presente capítulo detalla cada uno de los puntos fundamentales dentro del diseño e implementación de la propuesta de solución, tales como la arquitectura del sistema, los patrones de diseño utilizados, las funcionalidades y estructura de clases del sistema desarrollado.

Capítulo 3: Pruebas de software y resultados

Este capítulo está dedicado a la elaboración, ejecución y análisis de los resultados de los casos de prueba que evalúan las funcionalidades de la herramienta de configuración, antes de ser entregado el producto a la línea Sistemas Empotrados.

Capítulo 1. Fundamentación teórica

EL marco teórico de esta investigación se enmarca en las características de los PLC, su funcionamiento, principales componentes de hardware y los elementos configurables. También aborda algunos aspectos a tener en cuenta para la configuración de autómatas, así como ejemplos de aplicaciones creadas por fabricantes de PLC que realizan estas tareas. Otro aspecto importante tratado en esta sección es la definición de las tecnologías y herramientas que se emplean durante el desarrollo del trabajo.

1.1. Fundamentos de la automatización industrial

"En los últimos años los sistemas informáticos, mecánicos, electrónicos y de comunicaciones (redes y protocolos) se integran entre ellos en un todo armónico y funcional como un único complejo automático. Así nació la automatización que ya se ha convertido en el fundamento de todos los procesos industriales avanzados, y en consecuencia, una disciplina de base común a todas las direcciones de especialización profesional. La automatización en su concepto más amplio, es el control y la gestión de sistemas automáticos accionados mediante un conjunto de técnicas y dispositivos particulares"(1).

Su estudio y aplicación ha contribuido al reconocimiento universal de sus ventajas y beneficios asociados al ámbito industrial, que es donde tiene una de sus mayores aplicaciones debido a la necesidad de controlar un gran número de variables y por la creciente complejidad de los sistemas industriales (2).

1.2. Los Controladores Lógicos Programables o PLC

1.2.1 Definición de un PLC

"Se entiende por PLC, también conocido como autómata programable, a toda máquina electrónica diseñada para controlar en tiempo real y en medio industrial procesos secuenciales" (3).

Esta definición se ha quedado un poco desfasada porque han aparecido los micro-PLC, que no son más que PLC basados en microprocesadores o microcontroladores, destinados a necesidades menos complejas.

De acuerdo con la definición de la National Electrical Manufacturers Association (NEMA) un PLC es: "Un aparato electrónico operado digitalmente, que usa una memoria programable para el almacenamiento interno de instrucciones para implementar funciones específicas, tales como lógica, secuenciación, registro y control de tiempos, conteo y operaciones aritméticas para controlar, a través de módulos de entrada/salida (E/S) ya sean digitales o analógicos varios tipos de máquinas o procesos".

Este dispositivo realiza la acción de control en un proceso automático. El mismo actualiza constantemente su salida sobre el elemento de acción final o actuador, comparando el valor deseado para la variable a controlar, con el valor de la salida del proceso, proveniente del sensor (4). Tanto el dispositivo programable como sus periféricos asociados, están diseñados para ser integrados fácilmente dentro de un sistema de control industrial (5).

1.2.2 Los microcontroladores

"El microcontrolador es un circuito integrado programable que contiene todos los componentes de un computador: unidad central de procesamiento, memoria y unidades de E/S. Disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como conversores análogo digitales, temporizadores y buses de interfaz serie especializados" (6).

La labor del usuario consiste en realizar el programa**,** donde se define y ejemplifica la relación entre las señales de entrada que se tienen que cumplir para activar cada salida. La aplicación que permite la programación de PLC es específica para cada uno, de ahí que cada fabricante de PLC la proporcione junto al dispositivo. Esta aplicación consta básicamente de dos partes fundamentales: una que integra el software que permite la programación del PLC y otra de hardware, donde se realiza la configuración de los aspectos necesarios para que el dispositivo pueda operar en el ambiente industrial deseado.

1.2.3 Arquitectura o estructura de los PLC

Para realizar una correcta programación y conseguir un montaje y puesta en funcionamiento perfectos de un PLC, el mismo no se debe ver como una caja negra y

conocerlo tal cual es, como un equipo electrónico complejo montado en tarjetas específicas que controlan áreas o bloques, realizando distintas funciones que unidas convenientemente da como resultado a los PLC (7).

Estructura interna

Según los autores Jack Hugh (8) y el ingeniero Norberto Molinari (3), la estructura interna del autómata programable consta esencialmente de los siguientes bloques:

- Módulo de entradas.
- La unidad central de procesos o CPU.
- Módulo de salidas.
- La unidad o fuente de alimentación.
- La unidad o consola de programación si no se programa desde el ordenador (PC).
- Los dispositivos periféricos.
- Interfaces.

Módulo de entradas: Adapta y codifica de forma comprensible por la CPU las señales procedentes de los dispositivos de entrada o captadores (interruptores, finales de carrera, sensores, etcétera). Además tiene la misión de protección de los circuitos electrónicos internos del autómata, realizando una separación eléctrica entre estos y los captadores.

Módulo de salidas: Trabaja en forma inversa a las de entradas, es decir, decodifica las señales procedentes de la CPU, las amplifica y manda con ellas los dispositivos de salida o actuadores (lámparas, relés, contactores, arrancadores, electroválvulas). En este módulo también existen unas interfaces de adaptación a las salidas y de protección de circuitos internos.

CPU: Ejecuta todas las operaciones lógicas y/o aritméticas que requiere el controlador, realizadas por los microprocesadores. Básicamente es la encargada de procesar el programa de usuario, es decir, de interpretar las instrucciones y gobernar las salidas deseadas en función del valor de las entradas en cada momento (8).

Fuente de alimentación: Convierte la tensión de la red al valor que se utiliza como tensión de trabajo en los circuitos electrónicos internos que forma el autómata y de los dispositivos de E/S.

Dispositivos periféricos: Son aquellos elementos auxiliares, físicamente independientes del autómata, que unen al mismo para realizar su función específica y que amplían su campo de aplicación o facilitan su uso.

Interfaces: Son aquellos circuitos o dispositivos electrónicos que permiten la conexión a la CPU de los elementos periféricos discretos.

1.3. Configuración de PLC

Es necesario profundizar en algunos aspectos fundamentales del funcionamiento de un PLC para comprender la configuración del mismo.

La Unidad Central de Procesos está constituida por los elementos siguientes: procesador, memoria y circuitos auxiliares asociados (9).

Procesador

En el procesador radica principalmente el microprocesador y es fundamental aclarar que el aprovechamiento de la capacidad de un microprocesador está dado por el sistema operativo, el cual es un componente básico del controlador programable. Dos fabricantes de PLC pueden usar el mismo microprocesador con diferentes sistemas operativos, lo que determinará distintas características para cada equipo.

Una CPU con microprocesador es capaz de realizar cuatro tipos básicos de operaciones:

1. Aritméticas y lógicas tales como suma, resta, AND, OR, etcétera.
2. Operaciones de saltos que hacen posible pasar de una posición a otra de un programa.
3. Operaciones de lectura y modificación de contenidos de memoria.
4. Operaciones de E/S que hacen que el sistema pueda comunicarse con el mundo exterior.

Es necesario hacer una distinción entre las instrucciones usadas para comandar al microprocesador (programa ejecutivo o sistema operativo) y las instrucciones utilizadas por el programador para tratar un problema específico de control (Programa de aplicación del usuario).

El programa ejecutivo o sistema operativo es diseñado por el fabricante y normalmente no es accesible para el programador de la aplicación. El sistema operativo aprovecha la capacidad general de computación del microprocesador convirtiéndolo en una aplicación especializada del controlador lógico programable (9).

Memoria del PLC

El área de memorias sirve para dar alojamiento al programa ejecutivo o sistema operativo, programa de aplicación, tablas de datos y área auxiliar.

En general la demanda de memorias se divide en dos grandes grupos (9):

- Datos del proceso
    - Señales de planta, entradas y salidas
    - Variables internas
    - Datos alfanuméricos y constantes

- Datos del control
    - Programa del usuario
    - Configuración del PLC (cantidad de entradas/salidas conectadas, modo de funcionamiento, etcétera.)

Para dar respuesta a estas demandas, los controladores hacen uso de distintos tipos de memorias según sea su capacidad de almacenamiento, su velocidad de acceso, su volatilidad, etcétera.

1.3.1 Herramientas de configuración de PLC

Luego de un estudio sobre diferentes manuales de configuración, guías de usuario e instalación de herramientas existentes que configuran PLC se resume lo siguiente:

Las aplicaciones y herramientas para realizar las configuraciones necesarias sobre los PLC, brindan variadas opciones de asistentes que guían y dirigen este proceso. Incorporan módulos pre elaborados donde en muchas ocasiones las modificaciones son mínimas. Ofrecen en su mayoría laboratorios, ejemplos y manuales de ayuda al usuario.

Realizar la configuración generalmente suele ser engorrosa en la primera ocasión, pero una vez que se logra poner en funcionamiento al PLC, no es necesario volver a realizarla, al no ser en casos donde ocurran cambios considerables en el proceso que afecten directamente las configuraciones.

Existen diferentes elementos configurables en los PLC, estos dependen del tipo de hardware que presenten. La comunicación es uno de estos elementos ya que las formas en que los PLC intercambian datos con otros dispositivos son muy variadas. Típicamente un PLC puede tener integrado puertos de comunicaciones que pueden cumplir con distintos estándares de acuerdo al fabricante. Estos puertos pueden ser de los siguientes tipos:

- RS-232
- RS-485
- RS-422
- Ethernet

Sobre estos tipos de puertos de hardware las comunicaciones se establecen utilizando algún tipo de protocolo o lenguaje de comunicaciones. En esencia, un protocolo de comunicaciones define la manera en que los datos son empaquetados y codificados para su transmisión. De estos protocolos los más conocidos son:

- Modbus
- Bus CAN
- Profibus
- Devicenet
- Controlnet
- Ethernet I/P

Muchos fabricantes además ofrecen distintas maneras de comunicar sus PLC con el mundo exterior mediante esquemas de hardware y software protegidos por patentes y leyes de derecho de autor.

Otros de los elementos que necesitan ser configurados son las E/S, variables de operación del CPU y la fuente de alimentación. Estos generalmente son opciones a elegir y realizar mínimas parametrizaciones ya que son muy específicos del hardware que presente el dispositivo. Los fabricantes en su mayoría facilitan la tarea de configuración, un ejemplo de ello es la posibilidad de encuestar el hardware del equipo, cuyo resultado posibilita obtener una gama de opciones a seleccionar y valores de parámetros asignados automáticamente por la aplicación, esto se pone en evidencia al seleccionar el tipo de CPU, la fuente de alimentación, la cantidad de entradas y salidas que soporta, entre otros.

Los módulos pre-elaborados de entradas y salidas (analógicas o digitales) que brindan muchas de estas aplicaciones constituyen una gran ventaja, ya que una vez definido el diseño del proceso basta con adaptarlo a uno de los módulos y realizar algún cambio sobre todo en algún parámetro de estas variables.

Cuando se pone en marcha el PLC se realizan una serie de comprobaciones:

- Funcionamiento de las memorias.
- Comunicaciones internas y externas.
- Elementos de E/S.
- Tensiones correctas de la fuente de alimentación.

Una vez efectuadas estas comprobaciones y si las mismas resultan ser correctas, la CPU inicia la exploración del programa que no es más que el ciclo de funcionamiento del mismo. El tiempo de exploración del programa es variable en función de la cantidad y tipo de las instrucciones así como de la ejecución de subrutinas. Este tiempo es configurable ya que es uno de los parámetros que caracteriza a un PLC y generalmente se suele expresar en milisegundos por cada mil instrucciones.

Los pasos básicos que se deben realizar para que un PLC quede configurado y programado son los siguientes:

- Conexión del autómata a la PC de programación.

Es en este paso donde la comunicación queda configurada

- Configuración de la CPU.
  Esta básicamente además de escoger el tipo de CPU, incluye seleccionar diferentes parámetros como el modo de funcionamiento, frecuencia, entre otros.
- Configuración de los elementos de E/S.
  En dependencia de la aplicación se seleccionan módulos establecidos o se adicionan y configuran.
- Creación del programa.
  Es donde se crea el programa que representa el diseño del proceso deseado en el editor seleccionado.
- Carga del programa en el autómata.
- Puesta en funcionamiento del programa.
  Puede hacerse con independencia de la PC, este es el modo de funcionamiento normal cuando el autómata está instalado como controlador de un proceso industrial con control desde la PC. Este será el modo de funcionamiento común en la fase de depuración del programa.
- Comprobación del funcionamiento del programa.

1.4. Ejemplos de herramientas para configurar y programar PLC

Mientras que los conceptos fundamentales de la programación del PLC son comunes a todos los fabricantes, las diferencias en el direccionamiento E/S, la organización de la memoria y el conjunto de instrucciones hace que los programas de los PLC nunca se puedan usar entre diversos fabricantes. Incluso dentro de la misma línea de productos de un solo fabricante, diversos modelos pueden no ser directamente compatibles.

Un ejemplo de ello es el fabricante Siemens, que ofrecen una gama de modelos de PLC cada uno con su diferente software de configuración y programación.

1.4.1 Siemens

Esta marca maneja los tres lenguajes de programación: Esquema de contactos, Diagrama de funciones y Lista de Instrucciones y posee herramientas poderosas y de variadas

opciones de configuración, pero a veces de difícil comprensión sin ayuda de manuales o cursos de preparación.

La serie S7-200, a la que el fabricante le llama mini autómata, fue pensada para resolver la mayoría de las tareas básicas de automatización. Estos PLC se programan exclusivamente con el software Step7 Micro/WIN.

Las series S7-300 y S7-400 fueron creadas para aplicaciones mucho más complejas y utilizan el software Step7 con sus diferentes versiones un ejemplo es Step7 v5.2. La marca Siemens es sin duda, de las más empleadas en la industria debido a su robustez (10).

Step7 Micro/WIN

El S7-200 es el PLC de gama baja de la línea SIMATIC. Un PLC que permite comunicaciones de todo tipo: MODBUS Master/Slave, comunicación GSM que permite enviar y recibir mensajes (SMS), PROFIBUS DP (esclavo únicamente), Ethernet con servidor Web/Servidor de Emails/FTP, y el modo Freeport para crear protocolos propios o adaptarlo a protocolos no estándar por ejemplo, para recibir datos de un lector de código de barras (11).

El software Step7 Micro/WIN brinda variadas opciones para la comunicación como son:(11)

- Indicar el tipo de comunicación empleada (cable conversor o tarjeta de comunicaciones) y su configuración.
- En la ventana de *'Propiedades'* se verifica la configuración de los parámetros tales como dirección, timeout, velocidad de transferencia, entre otros.

Una vez realizado este procedimiento la aplicación brinda la opción de verificar la conexión realizada y muestra una lista con los autómatas conectados (en caso de ser más de uno), la lista muestra además las propiedades de los autómatas.

Para la configuración de la CPU el software brinda la opción de realizar una consulta desde la PC hacia el autómata para averiguar cuál es el modelo exacto de CPU del que dispone el equipo y propondrá la configuración.

Para realizar la programación de los PLC el software incluye los tres leguajes de programación antes mencionados. Independiente del tipo de lenguaje escogido, el software permite arrastrar los elementos ya sean botones o componentes específicos de cada lenguaje hacia el editor, esto facilita mucho la programación, además posee una tabla de símbolos con los datos necesarios a guardar por el programa.

En el paso de comprobar del funcionamiento del programa una vez puesto en marcha brinda una forma de realizarlo desde un entrenador utilizando:

- Los interruptores, que actúan como entradas activadas manualmente.
- Los leds indicativos de salidas activas o inactivas integrados en el autómata.

Ventajas

Posee gran facilidad de uso, es estándar para Windows, parametriza en lugar de programar los asistentes, gran repertorio de instrucciones aplicables con solo arrastrar y colocar. Además posee funciones de estado para los lenguajes AWL[1],KOP[2] y FUP[3], librerías complementarias para el software que agilizan las tareas de configuración y programación. Es ampliable modularmente, posee muy buenas respuestas en tiempo real Es un software que se ha probado en disimiles procesos en el mundo con resultados satisfactorios.

1.4.2 ZelioSoft

Otro de los fabricantes de PLC reconocidos internacionalmente es Schenider. ZelioSoft es el software que configura y programa la marca de PLC ZelioLogic de este fabricante. Este equipo presenta dos gamas para elegir: compacta y modular(12).

El software ZelioSoft incluye:

- El software de programación.
- Un módulo de autoformación.
- Una biblioteca de aplicaciones.
- Instrucciones técnicas.

[1] Lista de instrucciones o Nemónicos (AWL)
[2] Esquema de contactos o Ladder (KOP)
[3] Diagrama de funciones (FUP)

Ventajas

La programación y parametrización puede adaptarse fácilmente a las necesidades del usuario por los diferentes módulos pre elaborado que brinda. Presenta una programación rápida y sencilla usando FUP o KOP.

El FUP ofrece una capacidad de procesamiento de hasta 200 bloques de función y 23 funciones pre programadas que facilitan esta tarea al usuario, funciones para los sistemas de automatización secuenciales y además 6 funciones lógicas.

El lenguaje Ladder, presenta símbolos eléctricos muy intuitivos para el programador, admite hasta 120 líneas de esquemas de control, posee diferentes funciones como retro iluminación programable, cambio de reloj automático (1 hora verano/invierno).

Contiene un simulador integrado para ver cómo funciona el programa usuario, lo que proporciona una mejor seguridad al código:

- Test de coherencia
  Al menor error de entrada, ZelioSoft cambia a rojo y localiza el problema con gran precisión.
- Modos de simulación y control
  Test del programa en tiempo real, con o sin relé programable conectado a la PC.
- Ventana de supervisión
  Permite ver los estados de las E/S del relé programable en su entorno de aplicación.

Ofrece una configuración fácil de ampliar haciendo uso de extensiones a la gama modular con simples clics.

1.4.3 Moeller

Este fabricante brinda una gama de productos PLC Easy. El software EasySoft Basic es utilizado para los Easy400/500, Easy600/700 y el EasySoft Pro soporta los Easy400/500, Easy600/700, Easy800 y MFD-Titán.

EasySoft presenta el esquema de contacto como único lenguaje de PLC, el mismo cumple con la norma internacional IEC[4] para describir y ejemplificar símbolos de las bobinas, poseer un solo lenguaje se considera una desventaja aunque este lenguaje es el nativo para PLC y uno de los más utilizados por los programadores de estos dispositivos. El software dispone de una ventana para controlar el flujo de corriente de las vías lógicas. Presenta modos del software en 13 idiomas incluido además en documentación. El software de programación EasySoft-Pro para Easy800 con un nuevo componente de configuración denominado SmartWire-DT, presenta módulos de ampliación en las entradas y salidas, la comunicación y otros importantes campos con el configurador SmartWire-DT que permite la configuración manual o automática (13).

Ventajas

Este software brinda diferentes facilidades al usuario tales como (14):

- El editor gráfico muestra directamente la representación del diagrama circuito deseado.
- El menú de selección y las funciones de arrastrar y soltar, permiten establecer los vínculos con los contactos y bobinas con simples clics.
- Soporta grandes aplicaciones interconectadas en red con varios cientos de E/S y visualización gráfica.
- La herramienta integrada off-line permite a los usuarios comprobar el funcionamiento del circuito antes de la puesta en marcha y sin un dispositivo conectado.
- Las conexiones de los circuitos ya sean en serie o en paralelo se crean sin grandes conocimientos de programación.
- Módulos pre confeccionados que se integran fácilmente a las bobinas y el cableado.
- Presenta una interfaz muy sugerente al usuario.

1.5. Herramientas y tecnologías a utilizar

[4]La **Comisión Electrotécnica Internacional** (CEI o *IEC*, por sus siglas del idioma inglés *International Electrotechnical Commission*) es una organización de normalización en los campos eléctrico, electrónico y tecnologías relacionadas. Numerosas normas se desarrollan conjuntamente con la ISO (normas ISO/IEC).

El desarrollo de software no fuera posible sin las herramientas y tecnologías que le dan soporte, pues de no existir estas el trabajo sería engorroso, tedioso y muy lento para la obtención de los resultados. Las metodologías y herramientas a utilizar para llevar a cabo la propuesta de solución son las siguientes:

1.5.1 Metodologías de Desarrollo de Software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software ya que definen: Quién debe hacerse, qué, cuándo y cómo debe hacerlo, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software. Se podrían clasificar en dos grandes grupos: (15)

- Metodologías pesadas.
- Metodologías ligeras/ágiles.

Metodologías pesadas

Son las más tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida.

Una de las metodologías pesadas más conocidas y utilizadas es la metodología RUP (Rational Unified Process).

Metodologías ágiles

Metodologías orientadas a la interacción con el cliente y al desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando.

Principios

Para comprender mejor las características de las metodologías ágiles y su diferencia con las tradicionales existen principios que resumen la esencia de esta metodología. Los principios son (15):

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- Las personas del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Dentro de las metodologías ágiles o ligeras más usadas está Extreme Programming o Programación Extrema (XP) donde el cliente forma parte del equipo de desarrollo, realizando pequeñas iteraciones y mini entregas cada dos semanas, donde no existe más documentación que el propio código. Permite realizar cambios rápidamente al producto y finalizarlo en un corto plazo de tiempo con la mayor calidad posible.

Extreme Programming

La programación extrema, por sus siglas en inglés (XP), es un enfoque de la ingeniería de software formulado por Kent Beck[5], autor del primer libro sobre la materia, Extreme Programming Explained en 1999, es la metodología más destacada de los procesos ágiles de desarrollo de software. Centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Esta se define como especialmente adecuada para proyectos con requisitos imprecisos, muy cambiantes, y donde existe un alto riesgo técnico (16).

Las características fundamentales de XP son: (17)

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.
- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes tan pequeñas como sea posible.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el

[5]**Kent Beck** es ingeniero de software estadounidense, uno de los creadores de las metodologías de desarrollo de software de programación extrema y el desarrollo guiado por pruebas (Test-Driven Development o TDD), también llamados metodología ágil. Beck fue uno de los 17 firmantes originales del Manifiesto Ágil en 2011 .

personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.
- Semanas de 40 horas: XP lleva a un modo de trabajo en que el equipo está siempre al 100%. Una semana de 40 horas en las que se dedica la mayor parte del tiempo a tareas que suponen un avance puede dar mucho de sí, y hace innecesario recurrir a un esfuerzo mayor, excepto en casos extremos.

  Además, el sobreesfuerzo continuado pronto lleva a un rendimiento menor y a un deterioro de la moral de todo el equipo.
- Estándares de codificación: para conseguir que el código se encuentre en buen estado y que cualquier persona del equipo pueda modificar cualquier parte del código es imprescindible que el estilo de codificación sea consistente. Un estándar de codificación es necesario para soportar otras prácticas de la XP.
- Uso de metáforas: la comunicación fluida es uno de los valores más importantes de XP, para conseguir que esto ocurra es imprescindible, entre otras cosas, utilizar el vocabulario del negocio. También es fundamental huir de definiciones abstractas. Dicho de otro modo, la XP no pretende seguir la letra de la ley, sino su espíritu. Dentro de este enfoque es fundamental buscar continuamente metáforas que comuniquen intenciones y resulten descriptivas, enfatizando el qué por delante del cómo.

Se deben cumplir con todas estas características para que XP resulte una metodología eficiente que mejore el ciclo de desarrollo del software.

Para el desarrollo de la aplicación se decidió por todas las características antes mencionadas utilizar la metodología de desarrollo de software XP.

1.5.2 UML

El Lenguaje Unificado de Modelado por sus siglas en inglés (UML) “es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software” (18).

UML es el lenguaje que permite la modelación de sistemas de software con tecnología orientada a objetos, es uno de los más conocidos y utilizados en la actualidad. Se utiliza para definir un sistema en cada una de las etapas por las que tiene que pasar, indica que es lo que supuestamente hará, pero no como lo hará. Incluye aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Por lo que es muy útil para comprender el diseño del sistema (18).

Para la solución propuesta se utilizará el nuevo estándar: UML 2.0.

1.5.3 Las herramientas CASE: Visual Paradigm

Las Herramientas de Ingeniería de Software Asistida por Ordenador, por sus siglas en inglés (CASE),"son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras"(19).

Para el modelado de los artefactos y diagramas generados a lo largo del ciclo de vida del proyecto se decidió emplear Visual Paradigm en su versión 3.4, pues su uso está muy estandarizado a nivel mundial y constituye una herramienta multiplataforma muy madura y acabada.

Visual Paradigm para UML es una herramienta profesional fácil de utilizar, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a la rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, la realización de ingeniería tanto directa como inversa, generar el código desde diagramas y generar la documentación automáticamente en varios formatos como Web o Formato de Documento Portable (pdf) y el control de versiones. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona también abundantes tutoriales de UML, demostraciones interactivas de UML

y proyectos UML. La única desventaja es que es una herramienta propietaria, pero distribuyen copias gratuitas para la educación (19).

1.5.4 Lenguaje de programación utilizado: C++

"Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina" (20).

C++ es un lenguaje de programación diseñado con la intención de extender al exitoso lenguaje de programación C con mecanismos que permitiesen la manipulación de objetos. Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que C++ es un lenguaje multiparadigma. C++ permite trabajar tanto a alto como a bajo nivel (21).

Entre las principales ventajas de C++ como lenguaje de programación orientado a objetos se encuentran:

- C++ es un lenguaje de propósito general, o sea, que con él se puede programar cualquier cosa, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos y procesadores de texto.
- Los programas escritos en C++ tienen la ventaja de que podrán ejecutarse en cualquier máquina y bajo cualquier sistema operativo.
- Es un lenguaje multinivel, es decir, se puede usar para programar directamente el hardware o para crear aplicaciones tipo Windows.
- Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

1.5.5 Técnica de programación: Programación Orientada a Objetos

Con el uso de esta técnica de programación se pretende simplificar la modificación y extensión del software para lograr una mayor reutilización del mismo, permite una fácil comprensión debido a que la estructura del software y el problema a resolver están relacionados directamente. Aplicando diseño orientado a objetos, se crea software

resistente al cambio y escrito con economía de expresión. Esta técnica es la más usada actualmente por programadores (22).

1.5.6 Marco de trabajo: Qt

Qt es un marco de trabajo (framework) escrito en C++ para el desarrollo de aplicaciones de interfaz gráfica de usuario, el cual sigue la filosofía de software “escriba una vez, compile donde quiera” (write once, compile any where). Además, amplía las posibilidades del lenguaje C++ con una extensa biblioteca de clases, de uso intuitivo y dividida en módulos, en los cuales se puede encontrar desde programación de interfaces gráficas de usuarios hasta programación de redes, procesamiento de textos enriquecidos, acceso a datos almacenados en varios de los gestores de bases de datos más populares. Incorpora herramientas para escribir aplicaciones robustas y en el menor tiempo posible como el Diseñador Qt (QtDesigner). A esto se le suma su extensa base de datos de ejemplos listos para usar. Qt está disponible bajo licencia LGPL, permitiendo el desarrollo de Software Libre, y si se desea, software comercial no libre, en ambos casos sin verse obligado al pago de licencias o derechos (23).

1.5.7 Sistema Operativo GNU/Linux

Software libre es la denominación del software que brinda libertad a los usuarios sobre un producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente; de modo más preciso, se refiere a cuatro libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie (para la segunda y última libertad mencionadas, el acceso al código fuente es un requisito previo). Una distribución de Linux es una variante de ese sistema operativo (SO) que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios (24).

La herramienta será desarrollada a partir de las posibilidades y recursos que brinda el software libre. Buscando la independencia y soberanía tecnológica que se obtienen de las ventajas de uso.

1.5.7.1 Distribución de Linux: Ubuntu GNU/Linux

Ubuntu es un sistema operativo mantenido por Canonical y la comunidad de desarrolladores. Utiliza un núcleo Linux, y su origen está basado en Debian. Ubuntu está orientado en el usuario promedio, con un fuerte enfoque en la facilidad de uso y mejorar la experiencia de usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto. Ubuntu es conocido por su facilidad de uso y las aplicaciones orientadas al usuario final. El sistema incluye además funciones avanzadas de seguridad.

Se decidió usar Ubuntu porque es una distribución de GNU/Linux de desarrollo muy estable, por lo que los paquetes que se desarrollen en él y quieran ejecutarse utilizando cualquier distribución siempre serán estables.

1.5.8 Entorno Integrado de Desarrollo: QtCreator

Los Entornos Integrados de Desarrollo, IDE (del inglés: Integrated Development Environment) constituyen programas compuestos por un conjunto de herramientas para los programadores que facilitan la escritura de programas. Pueden dedicarse en exclusivo a un sólo lenguaje de programación o bien, pueden utilizarse para varios. Los IDE facilitan un ambiente de trabajo amigable. Para el desarrollo de la aplicación se decidió usar QtCreator como IDE. QtCreator ofrece un ambiente de desarrollo completo para la creación de aplicaciones Qt. Es una herramienta ligera y multiplataforma, con un enfoque estricto hacia las necesidades de los desarrolladores de aplicaciones Qt.

Las principales características son: (23)
1. El editor avanzado de C++, para escribir, editar y navegar por el código fuente sin utilizar el ratón (mouse).
2. Interfaz gráfica para la depuración, que incrementa la percepción de la estructura de clases de Qt.

3. Integración con QtDesigner para editar los archivos de interfaz gráfica de usuario.

4. La herramienta localizador (Locator) para la navegación rápida por archivos de proyecto, funciones, clases e informaciones de ayuda.

5. Integración con QtHelp, facilitando el acceso a la ayuda de la API Qt (tipos de datos, funciones) de Qt a través de una ayuda sensible al contexto.

Capítulo 2. Propuesta del sistema. Diseño e implementación

El diseño del software es una etapa importante dentro de su ciclo de desarrollo, toda aplicación debe estar soportada por un adecuado diseño y una correcta selección de su arquitectura. El presente capítulo detalla cada uno de los puntos fundamentales dentro del diseño e implementación de la propuesta de solución, tales como la arquitectura del sistema, los patrones de diseño utilizados, las funcionalidades y estructura de clases del sistema.

2.1 Propuesta del sistema

La línea Sistemas Empotrados, como principal cliente, tiene como propósito que se desarrolle una herramienta informática capaz de configurar los parámetros de ejecución del PLC-HMI para que el mismo pueda ser utilizado en diferentes ambientes industriales según las necesidades de cada proceso.

Por “configurar” y “parametrizar” se entiende la forma de elegir los módulos que intervendrán en el proceso a controlar, así como los valores que tomarán los elementos del hardware del dispositivo seleccionado. Esta herramienta brindará una interfaz para la gestión de dichos elementos configurables, estos elementos son: variables de entrada y salida ya sean digitales o analógicas, puertos de comunicación entre los que se encuentran RS-232 y Ethernet, variables de operación del CPU como son la frecuencia, niveles y la memoria RAM.

La aplicación brindará la opción de salvar en archivos XML, local o directamente en el dispositivo las configuraciones realizadas. Además ofrece la posibilidad de cargar estos archivos ya sea de procedencia local o del dispositivo, en caso de ser remota, a través de un protocolo de comunicación.

Se propone como nombre para la herramienta de configuración que se desea construir: Apus-PLC.

Figura 1: Propuesta de la herramienta de configuración

Protocolo de comunicación

Se han desarrollado diferentes familias de protocolos para la comunicación por red de sistemas UNIX, el más usado es el TCP/IP. Este se refiere a dos protocolos de red, que son: *Transmission Control Protocol* (Protocolo de Control de Transmisión) e *Internet Protocol* (Protocolo de Internet) respectivamente. Estos protocolos pertenecen a un conjunto mayor de protocolos., dicho conjunto se denomina *suite TCP/IP*.

Los diferentes protocolos de la suite TCP/IP trabajan conjuntamente para proporcionar el transporte de datos dentro de Internet, Intranet o el sistema diseñado. En otras palabras, hacen posible que accedamos a los distintos servicios de la red. Estos servicios incluyen: transmisión de correo electrónico, transferencia de ficheros, grupos de noticias, acceso a la World Wide Web, etcétera.

Hay dos clases de protocolos dentro de la suite TCP/IP que son: protocolos a nivel de red y protocolos a nivel de aplicación.

Protocolos a nivel de red

En la solución propuesta se utilizarán los protocolos a nivel de red que propone la suite TCP/IP para la comunicación de Apus-PLC con el dispositivo PLC-HMI.

Estos protocolos se encargan de controlar los mecanismos de transferencia de datos. Normalmente son invisibles para el usuario y operan por debajo de la superficie del sistema. Dentro de estos protocolos tenemos:

- **TCP**. Controla la división de la información en unidades individuales de datos (llamadas paquetes) para que estos paquetes sean encaminados de la forma más eficiente hacia su punto de destino.
- **IP**. Se encarga de repartir los paquetes de información enviados entre el ordenador local y los ordenadores remotos, garantizando que los datos se encaminen al destino correcto.

Protocolos a nivel de aplicación

Para el nivel de aplicación se creará un protocolo que se adecue específicamente a las necesidades de la herramienta, utilizando algunas funcionalidades que propone el marco de trabajo Qt.

Los protocolos de este nivel son visibles para el usuario en alguna medida, un ejemplo de esto es el que se utiliza en la solución propuesta; donde el usuario solicita una carga o descarga con otro dispositivo para recibir o transferir un fichero (en dependencia de la solicitud del usuario), la conexión se establece, y comienza la transferencia. Durante dicha transferencia, es visible parte del intercambio entre la máquina del usuario y la máquina remota (mensajes de error o de confirmación de la transferencia, ficheros XML).

Para que los datos viajen al destino indicado y puedan ser recibidos de la forma correcta, se “arma” el paquete que contiene la información a enviar, donde los datos son encapsulados con un identificador que proporciona Qt, además al inicio de estos se adiciona un número que representa la cantidad de caracteres (Nc) que contiene el fichero de los datos (XML). El siguiente ejemplo muestra como son enviados los paquetes que contienen los datos a través de la red utilizando el protocolo de comunicación explicado anteriormente:

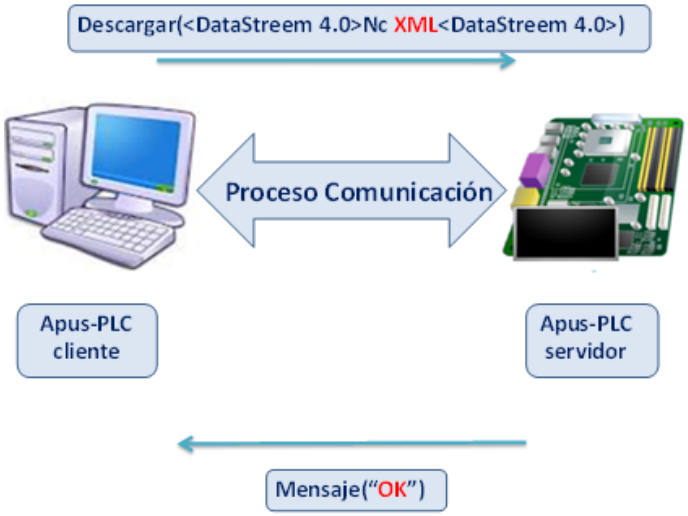


Figura 2. Estructura del paquete de información

2.2 Fase de exploración

La metodología de desarrollo Extreme Programming comienza con la fase de exploración. Durante esta, se realiza el proceso de identificación de las historias de usuario y características no funcionales del sistema, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del proyecto(24).

2.2.1 Características no funcionales del sistema

Las características no funcionales especifican las propiedades o cualidades que debe tener la solución a desarrollar. Representan las características que hacen al producto atractivo, rápido o confiable. Estos requisitos pueden marcar la diferencia entre un producto bien aceptado y otro con poca aceptación. A continuación se listan estas características: (25)

Características de usabilidad

- Facilidad de uso por parte del usuario.

Características de apariencia o interfaz externa

- La interfaz debe ser de fácil comprensión en su funcionamiento, permitiendo la utilización del sistema sin mucho entrenamiento.
- La interfaz debe contar con menús desplegables que faciliten y aceleren su utilización.
- Interfaces uniformes y con los mismos colores y diseños.
- Se debe garantizar que los colores de la interfaz de la aplicación sean claros.
- Mensajes sin ambigüedades.

Características de restricciones en el diseño e implementación

- El sistema debe ser desarrollado en el lenguaje de programación C++.
- Las interfaces gráficas de usuarios deben ser desarrolladas utilizando el marco de trabajo Qt.
- El sistema debe cumplir con las especificaciones técnicas del hardware del dispositivo PLC-HMI.

Características de ayuda y documentación

- El proyecto debe contar con una documentación según la metodología establecida.

Características Político culturales

- Debe respetar los términos empleados normalmente, por los especialistas en el tema de la esfera que se automatiza.

Características de portabilidad

- El sistema debe funcionar en sistemas de la familia GNU/Linux y Windows.

Características de hardware

La PC cliente:

- Procesador: 400 MHz
- Memoria RAM: 128 MB.
- Disco Duro: 25 MB.

2.2.2 Historias de usuario

"Las historias de usuario son la forma en que se especifican en XP los requisitos del sistema. Estas se escriben desde la perspectiva del cliente aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. El contenido de estas debe ser concreto y sencillo"(24).

Un punto estimado representará una semana ideal de trabajo como propone la metodología XP (8 horas durante 5 días). Se identificaron tres historias de usuario las cuales se detallan a continuación:

Tabla 1. Salvar y cargar configuración local

Historia de usuario	
Número: 1	**Usuario:** Línea Sistemas Empotrados
Nombre historia: Salvar y cargar configuración local.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Lauren San Juan	
Descripción: Permite salvar y cargar el archivo XML que contiene los parámetros de configuración de un dispositivo.	
Observaciones:	

Tabla 2. Salvar y cargar configuración remota

Historia de usuario	
Número: 2	**Usuario:** Línea Sistemas Empotrados
Nombre historia: Salvar y cargar configuración remota del PLC-HMI.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Lauren San Juan	
Descripción: Permite cargar del dispositivo PLC-HMI y salvar en el mismo, el archivo XML que contiene los parámetros de configuración que permiten su funcionamiento en un sistema de control industrial.	
Observaciones:	

Tabla 3. Gestionar los parámetros de configuración del dispositivo

Historia de usuario	
Número: 3	**Usuario:** Línea Sistemas Empotrados
Nombre historia: Gestionar los parámetros de configuración del dispositivo.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Lauren San Juan	
Descripción: Permite adicionar, modificar y visualizar los valores de los elementos de configuración del dispositivo.	
Observaciones:	

2.3 Fase de planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario, para ello se elaboran las tareas de ingeniería necesarias para desarrollar dichas historias, con una estimación de tiempo para cada funcionalidad, además se define el plan de iteraciones que explica qué implementar en cada iteración hasta obtener la versión final del producto.

2.3.1 Tareas de ingeniería

Tabla 4. Diseño e Implementación de la clase Dispositivo

	Tarea
Número tarea: 1	**Número historia:** 1
Nombre tarea: Diseño e Implementación de la clase Dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3/5
Fecha inicio: 16/01/2012	**Fecha fin:** 18/01/2012
Programador responsable: Lauren San Juan	
Descripción: La clase dispositivo es la que posee los parámetros de configuración de un dispositivo de control.	

Tabla 5. Estudio del módulo QtXml

	Tarea
Número tarea: 2	**Número historia:** 1
Nombre tarea: Estudio del módulo QtXml	
Tipo de tarea: Investigación	**Puntos estimados:** 2/5
Fecha inicio: 19/01/2012	**Fecha fin:** 20/01/2012

Programador responsable: Lauren San Juan
Descripción: El módulo de QtXml proporciona un lector de flujo y escritor de documentos XML, además de implementaciones en C++ de DOM.

Tabla 6. Diseño de la interfaz para salvar la configuración del dispositivo

	Tarea
Número tarea: 3	**Número historia:** 1
Nombre tarea: Diseño de la interfaz para salvar la configuración del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1/5
Fecha inicio: 23/01/2012	**Fecha fin:** 23/01/2012
Programador responsable: Lauren San Juan	
Descripción: La aplicación presenta un menú desplegable que contiene la opción de Salvar Proyecto y muestra una ventana para elegir la ubicación del archivo XML.	

Tabla 7. Salvar la configuración local del dispositivo

	Tarea
Número tarea: 4	**Número historia:** 1
Nombre tarea: Salvar la configuración local del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4/5
Fecha inicio: 24/01/2012	**Fecha fin:** 27/01/2012
Programador responsable: Lauren San Juan	
Descripción: Después de creado el archivo XML, se guarda en el mismo la configuración que presenta la clase Dispositivo.	

Tabla 8. Diseño de la interfaz para cargar la configuración del dispositivo

	Tarea
Número tarea: 5	**Número historia:** 1
Nombre tarea: Diseño de la interfaz para cargar la configuración del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1/5
Fecha inicio: 30/01/2012	**Fecha fin:** 30/01/2012
Programador responsable: Lauren San Juan	
Descripción: La aplicación presenta un menú desplegable que contiene la opción de Abrir Proyecto, la cual muestra una ventana para localizar el archivo XML que se desea cargar.	

Tabla 9. Cargar la configuración local del dispositivo

	Tarea
Número tarea: 6	**Número historia:** 1
Nombre tarea: Cargar la configuración local del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4/5
Fecha inicio: 31/01/2012	**Fecha fin:** 3/02/2012
Programador responsable: Lauren San Juan	
Descripción: Una vez localizado el archivo XML deseado, se van leyendo los datos y se almacenan en un objeto para que sean visualizados estos elementos.	

Tabla 10. Implementación del árbol de proyecto

	Tarea
Número tarea: 1	**Número historia:** 3
Nombre tarea: Implementación del árbol de proyecto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 6/02/2012	**Fecha fin:** 6/02/2012
Programador responsable: Lauren San Juan	
Descripción: Al elegir las opciones del menú Abrir Proyecto y Nuevo Proyecto, se crea un árbol de proyecto que permite seleccionar de un menú desplegable del dispositivo, los elementos configurables que se deseen adicionar o eliminar.	

Tabla 11. Implementación de los elementos configurables del dispositivo

	Tarea
Número tarea: 2	**Número historia:** 3
Nombre tarea: Implementación de los elementos configurables del dispositivo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 7/02/2012	**Fecha fin:** 13/03/2012
Programador responsable: Lauren San Juan	
Descripción: Implementar los componentes que permitirán visualizar, adicionar o modificar los valores de los elementos del dispositivo.	

Tabla 12. Implementación de la vinculación de los elementos configurables del dispositivo

	Tarea
Número tarea: 3	**Número historia:** 3
Nombre tarea: Implementación de la vinculación de los elementos configurables del dispositivo con el componente que contiene los valores de los mismos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14/02/2012	**Fecha fin:** 20/02/2012
Programador responsable: Lauren San Juan	
Descripción: Al dar clic izquierdo sobre algún despliegue del árbol de proyecto, se agregará un componente que permitirá visualizar, adicionar o modificar los valores del elemento seleccionado.	

Tabla 13. Diseñar e implementar la clase cliente

	Tarea
Número tarea: 1	**Número historia:**2
Nombre tarea: Diseñar e implementar la clase cliente	
Tipo de tarea: Desarrollo	**Puntos estimados:** 4/5
Fecha inicio: 21/02/2012	**Fecha fin:** 24/02/2012
Programador responsable: Lauren San Juan	
Descripción: Cliente es la clase que permitirá realizar la comunicación con el dispositivo desde la herramienta al lado de la PC-Cliente, además contendrá las funcionalidades que permitan cargar y descargar la configuración remota.	

Tabla 14. Implementar el protocolo de comunicación

	Tarea
Número tarea: 2	**Número historia:** 2
Nombre tarea: Implementar el protocolo de comunicación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27/02/2012	**Fecha fin:** 2/03/2012
Programador responsable: Lauren San Juan	
Descripción: Implementar el protocolo de comunicación que permita cargar y descargar archivos de configuración desde la herramienta al lado de la PC-Cliente hacia la herramienta al lado del servidor (dispositivo).	

Tabla 15. Diseño de la interfaz para cargar y descargar la configuración remota

	Tarea
Número tarea: 3	**Número historia:** 2
Nombre tarea: Diseño de la interfaz para cargar y descargar la configuración remota	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1/5
Fecha inicio: 5/03/2012	**Fecha fin:** 5/03/2012
Programador responsable: Lauren San Juan	
Descripción: La aplicación presenta un menú desplegable que contiene las opciones de Cargar y Descargar, ambas muestran una ventana para poner el IP del dispositivo. En la primera opción se carga en el árbol de proyecto la configuración remota y en la segunda opción de envía la configuración de la herramienta abierta en ese momento.	

Tabla 16. Diseñar e implementar la clase servidor

	Tarea
Número tarea: 4	**Número historia:** 2
Nombre tarea: Diseñar e implementar la clase servidor	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 6/03/2012	**Fecha fin:** 12/03/2012
Programador responsable: Lauren San Juan	
Descripción: Servidor es la clase que permitirá realizar la comunicación con la herramienta al lado de la PC-Cliente, además contendrá las funcionalidades que permitan enviar, recibir la configuración y los comandos necesarios para cambiar sus elementos de hardware por los recibidos.	

2.3.2 Plan de Iteraciones

Iteración 1

Esta iteración tiene como objetivo la implementación de algunas tareas de las historias de usuario que contienen funcionalidades que constituyen la base de la herramienta, por lo que durante el transcurso de la misma se crea la base de la arquitectura del sistema con funcionalidades mínimas, haciendo mayor énfasis en la implementación de salvas y cargas locales. Finalmente se obtiene una primera versión de prueba, la cual es mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Iteración 2

El objetivo de esta iteración es la implementación de las historias de usuario que presenten funcionalidades relacionadas a la gestión de los parámetros configurables, específicamente tareas que influyen directamente en los componentes visuales. Al finalizar se obtiene una versión de prueba que permite visualizar, insertar y modificar los elementos de configuración del PLC-HMI. Esta versión es mostrada al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

Iteración 3

Durante su transcurso se implementan las historias de usuario que poseen las funcionalidades de conexión con el dispositivo y se realizan las implementaciones necesarias para reutilizar las funcionalidades de las anteriores iteraciones e integrarlas a los procesos de conexión. Al terminar la iteración se obtiene la versión 1.0 del producto final, adicionando además aquellas funcionalidades que sirvan de enriquecimiento a la herramienta. Como resultado de la misma, el sistema es puesto en funcionamiento durante un período de tiempo para evaluar su desempeño.

Tabla 17. Plan de Implementación de las Iteraciones

Iteración	Orden de las historias de usuarios a implementar	Duración total de la iteración
Iteración 1	Salvar y cargar configuración local.	3 semanas
Iteración 2	Gestionar los parámetros de configuración del dispositivo.	3 semanas
Iteración 3	Salvar y cargar configuración remota del PLC-HMI.	3 semanas

2.4 Diseño e implementación del sistema

En esta fase del diseño se presenta una descripción de algunos elementos de la arquitectura teniendo en cuenta la metodología seleccionada para el desarrollo de la herramienta que se propone, estos son: diagrama general de clases del diseño, la descripción de las clases implicadas, los patrones de diseño aplicados entre otros.

2.4.1 Descripción de los elementos de la arquitectura

“La arquitectura es el esqueleto o base de una aplicación. Se necesita una arquitectura para comprender el sistema, organizar el desarrollo y hacer evolucionar el sistema” (25).

Estilos arquitectónicos

Un estilo arquitectónico define un conjunto de principios que le dan forma o rigen el diseño del sistema a desarrollar. Estos principios van encaminados a cómo interactúan y están estructurados los componentes o módulos del sistema, así como las responsabilidades de cada uno. La arquitectura de un sistema de software casi nunca está atada a un solo estilo arquitectónico, sino que es más bien una combinación de estos estilos los que dan como resultado la arquitectura final del sistema (25).

En el presente trabajo se asumió como principal estilo la arquitectura orientada a objetos, donde los componentes de un sistema encapsulan los datos y las operaciones que deben aplicarse para manipular los datos. La comunicación y coordinación entre componentes se consigue mediante el paso de mensajes (26).

Patrones arquitectónicos

Un patrón arquitectónico al igual que un estilo arquitectónico define un conjunto de principios para el diseño de la arquitectura, pero difieren de estos en que el alcance es menor, pues solo se concentran en un aspecto de la arquitectura (por ejemplo el manejo de la concurrencia y la distribución). Los patrones se usan en conjunto con un estilo arquitectónico para determinar la forma de la estructura general del sistema (27).

Arquitectura Cliente-Servidor

La tecnología Cliente-Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requisitos a uno o más servidores centrales. Este modelo provee usabilidad, interoperabilidad, flexibilidad y escalabilidad en las comunicaciones (28).

La herramienta de configuración está estructurada en dos partes: Apus-PLC del lado del cliente y Apus-PLC del lado del servidor. La arquitectura que presenta la herramienta para establecer las comunicaciones es la arquitectura Cliente-Servidor.

La comunicación entre el cliente y el servidor presenta dos escenarios posibles:

1. El cliente envía un mensaje al servidor solicitando cargar la configuración que presenta y este envía un mensaje que contiene el archivo XML correspondiente a su configuración. Como se muestra en la figura.

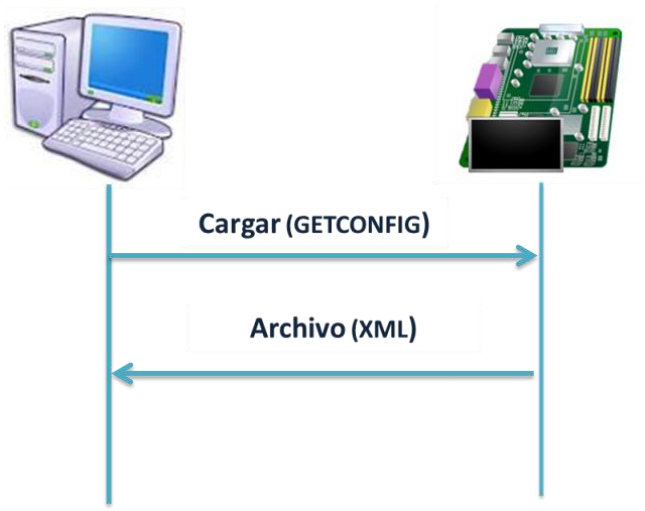


Figura 3: Comunicación escenario 1

2. El cliente envía un mensaje al servidor que contiene la configuración realizada y que este debe cargar y una vez recibida la configuración envía un mensaje de confirmación (en caso de haberse realizado el proceso satisfactoriamente). Como se muestra en la figura.

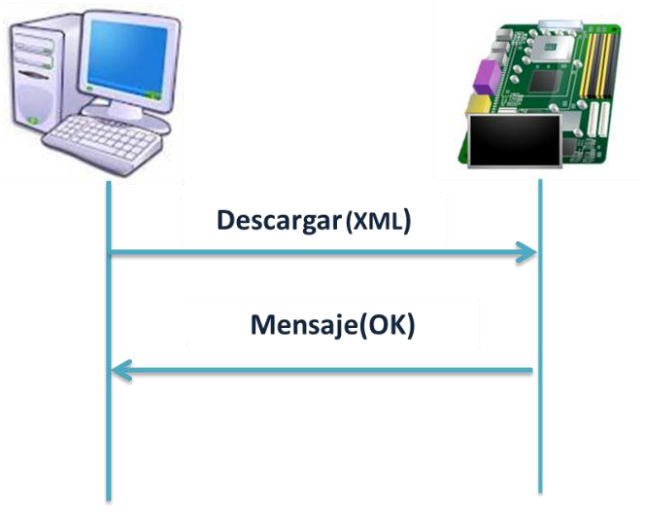


Figura 4: Comunicación escenario 2

Arquitectura Modelo/Vista

El patrón arquitectónico que representa la estructura de los datos y los componentes de la herramienta de configuración en sus dos partes, constituye una variación en el modo de funcionar del Modelo Vista Controlador (MVC).

El MVC divide una aplicación en tres componentes o roles principales (27):

- Modelo: es la representación específica de la información con la cual el sistema opera, es decir, sobre la cual funciona la aplicación. No depende de ninguna vista y de ningún controlador.
- Vista: maneja la visualización de la información. Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: controla el flujo entre la vista y el modelo(los datos). Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Es el encargado de realizar la lógica específica del negocio.

Si la vista y los objetos del controlador se combinan, el resultado es la arquitectura Modelo / Vista. Esta igual separa la forma en que se almacenan los datos, de la forma en que se presentan al usuario, pero lo proporciona en un marco más simple aunque basado en los mismos principios. Esta separación hace que sea posible visualizar los mismos datos en varias vistas diferentes, y poner en práctica nuevos tipos de vista, sin necesidad de cambiar las estructuras de datos.

El marco de trabajo Qt utilizado para el desarrollo de la herramienta de configuración, introduce un nuevo conjunto de clases de vista de elementos que utilizan un arquitectura Modelo / Vista para la gestión de la relación entre los datos y la forma en que se presenta al usuario. La separación de la funcionalidad introducida por esta arquitectura ofrece a los desarrolladores una mayor flexibilidad para personalizar la presentación de los elementos, y proporciona una interfaz estándar del modelo para permitir que una amplia gama de datos puedan ser utilizados con vistas de elementos ya existentes.

Además introduce un nuevo concepto: el uso de delegados para permitir un manejo más flexible de las entradas del usuario (29).

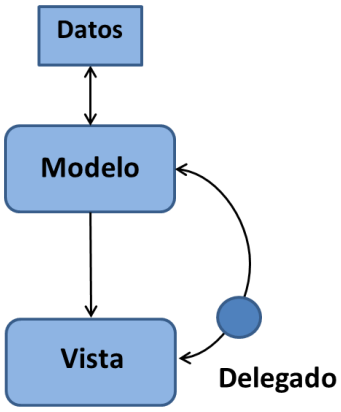


Figura 5: Patrón arquitectónico Modelo/Vista

Teniendo en cuenta las características mencionadas anteriormente y las funcionalidades ofrecidas por el marco de trabajo, se selecciona el patrón arquitectónico Modelo/Vista, a continuación se hace una descripción de los roles que se combinan en la solución propuesta:

- Vista: En este patrón la vista juega un rol como modelo de presentación, ya que contiene aquellas clases modelo que almacenan la lógica de presentación de los datos y maneja de forma independiente las clases que controlan el estado y las que rigen el comportamiento de una presentación (vista). Se utilizan delegados en los elementos que requieren una edición gráfica más compleja.
- Modelo: Se desempeña como modelo del dominio, contiene la información de los datos y la lógica del negocio.

La naturaleza de la comunicación depende del tipo de dato y la forma en que se implemente el modelo. Además las clases modelo de presentación, las vistas, y los delegados se comunican entre sí mediante señales y slots.

La siguiente figura muestra un diagrama de paquetes confeccionado con elementos de UML y esboza como queda organizada las dependencias entre los paquetes que representan cada uno de los roles antes mencionados.

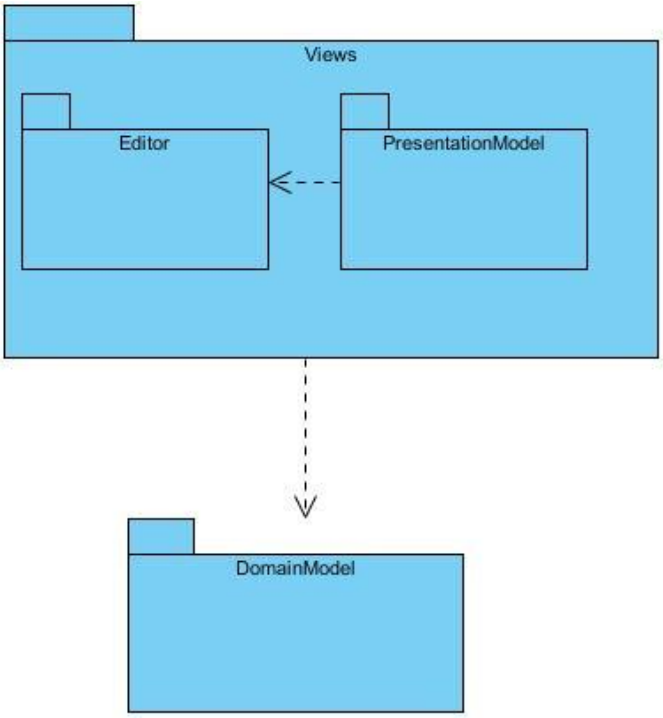


Figura 6: Vista de paquetes de la aplicación

Views

Editor: Clases que se encargan de visualizar los datos y manejar las interacciones del usuario con la aplicación.

PresentationModel: Clases que modelan la lógica de presentación de los datos de forma separada a la implementación de las vistas, permitiendo que los datos puedan ser presentados con vistas diferentes utilizando los mismos modelos.

DomainModel

Contiene el modelo de datos, la clase que representa al dispositivo y la relación del mismo con sus elementos configurables.

2.4.2 Definiciones de diseño que se aplican

Existen varias categorías o clasificaciones de patrones entre las que se encuentran los patrones arquitectónicos, patrones de diseños, entre otros. En esta sección se describen los patrones de diseños utilizados en la solución.

"Un patrón de diseño es una abstracción de una solución en un alto nivel que constituye la respuesta a un problema de diseño no trivial efectiva y reusable" (27).

Los patrones de diseño que se seleccionaron son los siguientes:

Patrones de diseño GRASP

Los Patrones Generales de Asignación de Responsabilidades, por sus siglas en inglés (GRASP), son patrones de diseño que se usan para asignar responsabilidades a una clase. Se pueden destacar 5 patrones básicos (principales) y 4 patrones adicionales a aplicar en el diseño Orientado a Objetos. A continuación se muestran los más utilizados en la solución propuesta, con una breve descripción y ejemplos donde son aplicados.

- Experto: Asignar una responsabilidad al más competente en información, la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos.

  Utilización: Este patrón se puede observar en la clase DeviceItem que contiene el dispositivo y es la encargada de crear sus elementos configurables, siendo capaz de validar todas las entradas posibles.

- Alta Cohesión: Asignar una responsabilidad de modo que la unión se mantenga a gran escala. Asignar a las clases responsabilidades que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño.

  Utilización: Se evidencia en la clase InputItem, que contiene un objeto de tipo InEditor encargado de editar las entradas y la colección de las mismas, trabajando únicamente sobre el área de entradas.

  Las clases de edición se encargan de visualizar los datos y manejar las interacciones del usuario, las clases de los modelos de presentación se encargan de realizar un modelo estándar de presentación de los datos y las clases de los modelos de dominios se encargan de contener los datos.

- Bajo Acoplamiento: Asignar una responsabilidad para mantener un engranaje pobre. Es un principio que se debe recordar durante las decisiones de diseño;.

Soporta el diseño de clases más independientes. Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.

Utilización: Se evidencia en la relación que tienen las clases CPU, CPUItem y CPUEditor, son las únicas relacionadas con el manejo de la CPU, siendo además las únicas relacionadas entre sí. Esta estructura permite que un cambio en otro elemento de configuración no afecte a estas clases

- Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Asigna la responsabilidad del manejo de mensajes de los eventos de un sistema a una clase.

  Utilización: Este patrón se evidencia en la clase Client, ya que sirve como intermediario entre una determinada interfaz y la comunicación con el servidor. En dependencia del método llamado invoca al algoritmo necesario para cumplir dicha funcionalidad.

- Indirección: Asignar la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios y estos no terminen directamente acoplados.

  Utilización: Este patrón se refleja en diferentes clases pues que actúan como intermediarios entre algunas vistas y los modelos del dominio, como por ejemplo InModel y OutModel.

2.5 Descripción de las clases

El diagrama de clases captura la estructura lógica del sistema y las clases que constituyen el modelo. Es un modelo estático, describiendo lo que existe y qué atributos y comportamientos tiene. Los diagramas de clases son los más útiles para ilustrar las relaciones entre las clases e interfaces (30).

El siguiente diagrama muestra la relación entre las clases pertenecientes al modelo del dominio y sus funcionalidades:

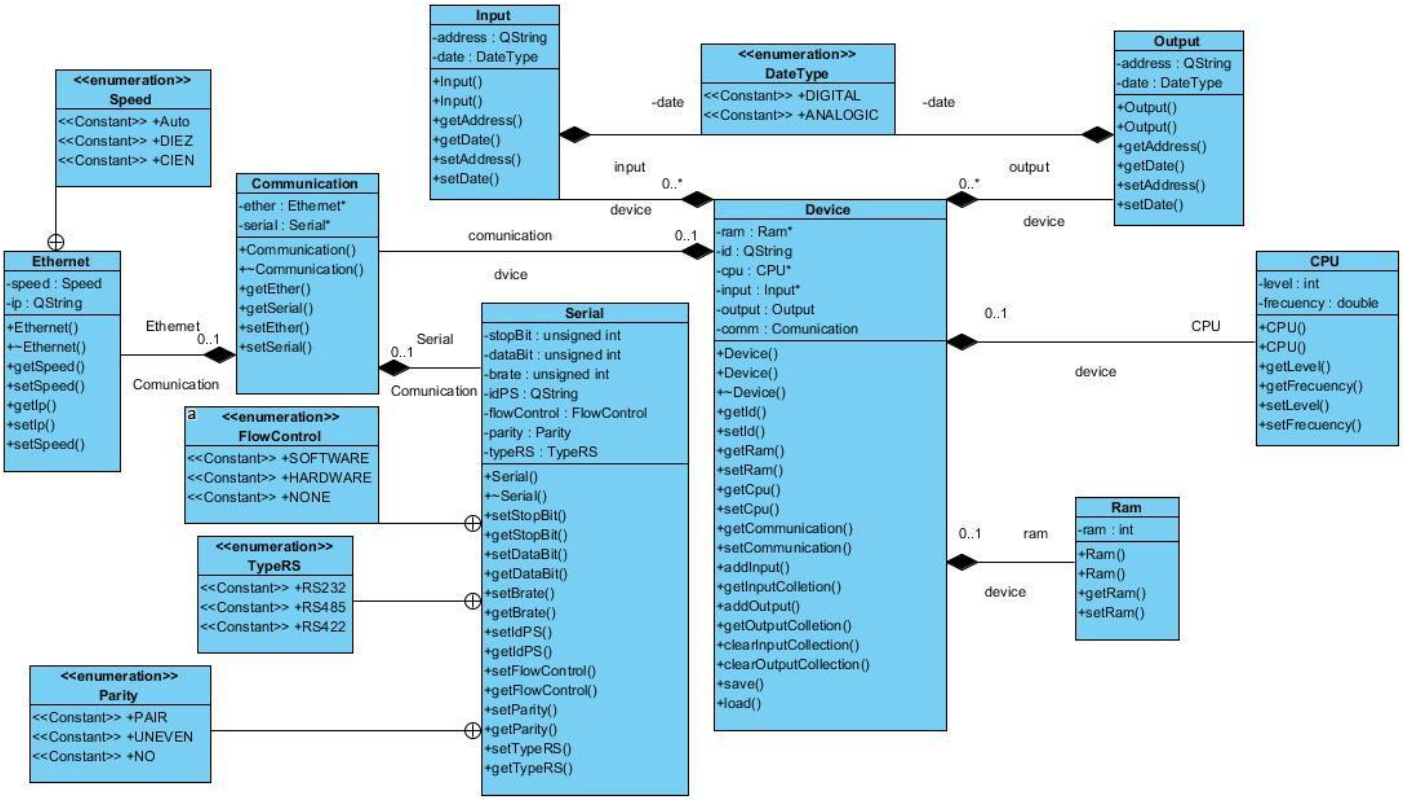


Figura 7: Diagrama de clases del modelo de datos

La metodología XP recomienda el uso de las tarjetas CRC (clase, responsabilidad y colaboración) son una técnica para el diseño de software orientado a objetos creada por Kent Beck y Ward Cunningham[6]. Permite ver las clases como algo más que un depósito de datos y conocer el comportamiento de cada una en un alto nivel.

La clase es cualquier persona, cosa, evento, concepto, vista, reporte, etcétera. Las responsabilidades de una clase son las cosas que conoce y que realiza (atributos y métodos). Los colaboradores son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

[6]**Ward Cunningham** es conocido por sus contribuciones en la práctica del desarrollo de programación orientada a objetos, patrones del diseño para el desarrollo de software especialmente bajo la metodología XP.

Tabla 18. Tarjeta CRC Device

Nombre: Device
Responsabilidades: Posee los parámetros del dispositivo (listado de variables de entrada y salida A/D, memoria, CPU, comunicación) y su gestión. Tiene además la responsabilidad de salvar y cargar la configuración del dispositivo. Es una entidad que contiene la lógica del negocio.
Colaboradores: Communication, CPU, Ram, Input/Output, InModel/OutModel.

Tabla 19. Tarjeta CRC DeviceItem

Nombre: DeviceItem
Responsabilidades: Posee todos los parámetros del dispositivo en forma de ítem para ser visualizados en el árbol de proyecto y los maneja de forma genérica. Contiene la lista de acciones que ejecutará el dispositivo ya sea de crear o eliminar los elementos. Tiene además la responsabilidad de leer los valores que contiene el dispositivo para presentarlos al usuario y mandar los archivos que serán guardados.
Colaboradores: Device, CPUItem, CommItem, InputItem, SerialItem, OutputItem, EthernetItem, ramItem, hereda de TreeWidgetItem.

Tabla 20.Tarjeta CRC Client

Nombre: Client
Responsabilidades: Contiene las funcionalidades necesarias para establecer la comunicación con el dispositivo, además maneja los mensajes entre ellos, como cargar y descargar la comunicación, los mensajes de confirmación de entrega de paquetes y la captura de posibles errores durante la trasmisión y la conexión. Esta clase se encuentra en la aplicación del lado del cliente.
Colaboradores: QtcpSocket, QtcpServer, QhostAddress, Qmutex, hereda de QObject

Tabla 21. Tarjeta CRC Server

Nombre: Server
Responsabilidades: Contiene las funcionalidades necesarias para permitir la comunicación con el servidor (dispositivo). Además las funcionalidades para recibir las peticiones del cliente como la de enviar la configuración (cargar) y descargar la configuración, así como enviar mensajes de confirmación de paquetes y mensajes de error durante la trasmisión o la conexión. Esta clase se encuentra en la aplicación del lado del servidor.
Colaboradores: QtcpSocket, QtcpServer, QhostAddress, hereda de QObject

Tabla 22. Tarjeta CRC ProyectTree

Nombre: ProyectTree
Responsabilidades: Es responsable de presentar los elementos del dispositivo en un árbol desplegable de proyecto.
Colaboradores: DeviceItem, hereda de QTreeWidget.

Tabla 23. Tarjeta CRC InModel

Nombre: InModel
Responsabilidades: Contiene la lista de variables de entrada, es la que construye el modelo de entradas que permite la gestión y visualización de sus elementos de forma independiente a las clases de presentación(clases de edición y clases ítem).
Colaboradores: Input, hereda de QStandardItemModel.

Tabla 24. Tarjeta CRC OutModel

Nombre: OutModel
Responsabilidades: Contiene la lista de variables de salida, es la que construye el modelo de salidas que permite la gestión y visualización de sus elementos de forma independiente a las clases de presentación (clases de edición y clases ítem).
Colaboradores: Output, hereda de QStandardItemModel.

Tabla 25. Tarjeta CRC TreeWidgetItem

Nombre: TreeWidgetItem
Responsabilidades: Es la clase genérica que representa un elemento dentro del árbol del proyecto, de ella heredan los elementos del dispositivo para ser visualizados en el árbol de proyecto.
Colaboradores: Hereda de QObjet y QtreeWidgetItem.

Tabla 26. Tarjeta CRC CPUEditor

Nombre: CPUEditor
Responsabilidades: Contiene los componentes gráficos que permiten mostrar y editar los parámetros del elemento CPU, es la que capta las instrucciones del usuario sobre dicho elemento.
Colaboradores: CPU, hereda de QWidget.

Tabla 27. Tarjeta CRC RamEditor

Nombre: RamEditor
Responsabilidades: Contiene los componentes gráficos que permiten mostrar y editar los parámetros del elemento RAM, es esta vista la capta las instrucciones del usuario sobre dicho elemento.
Colaboradores: Ram, hereda de QWidget.

Tabla 28. Tarjeta CRC InEditor

Nombre: InEditor
Responsabilidades: Contiene los componentes gráficos que permiten mostrar y editar los parámetros de las variables de entrada, es esta vista la capta las instrucciones del usuario sobre dichas variables.
Colaboradores: Input, hereda de QWidget.

Tabla 29. Tarjeta CRC OutEditor

Nombre: OutEditor
Responsabilidades: Contiene los componentes gráficos que permiten mostrar y editar los parámetros de las variables de salida, es esta vista la capta las instrucciones del usuario sobre dichas variables.
Colaboradores: Output, hereda de QWidget.

Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Este diagrama es útil para ilustrar la arquitectura física de un sistema (31).

A continuación se muestra una representación gráfica donde se encuentra la herramienta de configuración Apus-PLC situada en la PC cliente, conectada a través del protocolo de comunicación TCP/IP al dispositivo PLC-HMI que contiene la versión Apus-PLC en modo servidor.

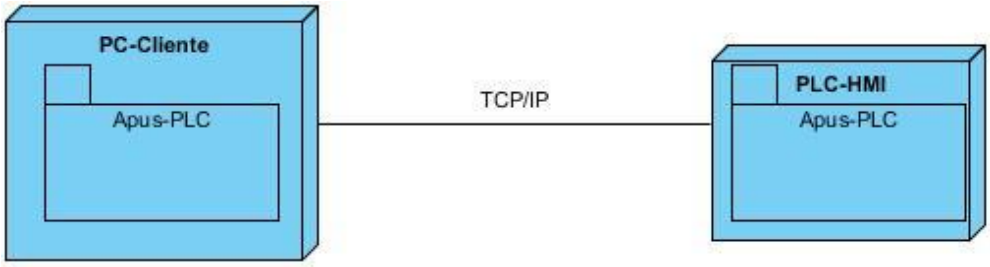


Figura 8: Diagrama de despliegue

Capítulo 3. Pruebas de software y resultados

Este capítulo está dedicado a la elaboración y ejecución de casos de prueba que evalúen las funcionalidades de la herramienta de configuración Apus-PLC, con el objetivo de encontrar la mayor cantidad de errores antes de ser entregado el producto a la línea Sistemas Empotrados como cliente principal. Se realizan las iteraciones necesarias para cumplir satisfactoriamente los casos de pruebas elaborados, obteniéndose finalmente los resultados de las pruebas realizadas.

3.1 Pruebas de software

La prueba de software es uno de los procesos fundamentales dentro del control de calidad del software. “Consiste en la ejecución de un programa bajo ciertos datos de entrada (casos de prueba), para posteriormente comparar las salidas obtenidas con las deseadas (función para la cual fue diseñado)” (32). La prueba de software puede intencionalmente forzar al programa a producir errores en las respuestas

El método de prueba que se aplica a la herramienta de configuración es el de caja negra; permitiendo examinar los aspectos funcionales del sistema haciendo mínimo enfoque en la estructura lógica interna del software. Esta técnica aplica pruebas de aceptación para verificar que el software está listo y que puede ser usado por los usuarios finales para ejecutar las funciones para el cual fue concebido. Este tipo de prueba es recomendado por la metodología seleccionada XP, donde en colaboración con el cliente se diseñan para cada historia de usuario

3.2 Diseño de casos de pruebas

Los casos se diseñaron a partir de las historias de usuario de la herramienta de configuración. Una historia de usuario puede tener tantos casos de prueba como sean necesarios para evaluar su funcionamiento.

A continuación se muestran los casos de prueba de aceptación elaborados para la herramienta de configuración:

Tabla 30. Caso de prueba 1

Caso de prueba de aceptación: 1
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Crear el dispositivo y sus elementos configurables en el árbol de proyecto.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de crear el dispositivo y sus elementos en el árbol de proyecto, funciona conforme a lo esperado.
Condiciones de ejecución:
Entradas/ Pasos de ejecución: • Seleccionar del menú la opción "*Nuevo Proyecto*". • Seleccionar de un menú desplegable del dispositivo el elemento a crear.
Resultado esperado: el dispositivo y sus elementos creados deben aparecer en el árbol de proyecto.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 31. Caso de prueba 2

Caso de prueba de aceptación: 2
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Eliminar del árbol de proyecto el dispositivo con todos sus elementos.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad eliminar el dispositivo y sus elementos, funciona conforme a lo esperado.
Condiciones de ejecución: Debe existir el dispositivo en el árbol de proyecto.
Entradas/ Pasos de ejecución: • Seleccionar la opción "Eliminar" del menú desplegable del dispositivo.
Resultado esperado: Se debe eliminar el dispositivo y todos sus elementos.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 32. Caso de prueba 3

Caso de prueba de aceptación: 3
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Eliminar elementos del dispositivo.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de eliminar elementos del dispositivo en el árbol de proyecto, funciona conforme a lo esperado.
Condiciones de ejecución: Debe existir el dispositivo con algún elemento en el árbol de proyecto.
Entradas/ Pasos de ejecución: • Seleccionar la opción “Eliminar” del despliegue del elemento deseado.
Resultado esperado: Se debe eliminar el elemento seleccionado y todos los parámetros que contenga.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 33. Caso de prueba 4

Caso de prueba de aceptación: 4
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Vincular los elementos del dispositivo con sus parámetros de configuración.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad que vincula los elementos del dispositivo con los parámetros de configuración, funciona conforme a lo esperado.
Condiciones de ejecución: Debe existir el dispositivo con algún elemento en el árbol de proyecto.
Entradas/ Pasos de ejecución: • Presionar doble clic sobre el elemento del dispositivo deseado.
Resultado esperado: Deben aparecer los parámetros de configuración del elemento seleccionado en la zona de trabajo de la aplicación.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 34. Caso de prueba 5

Caso de prueba de aceptación: 5
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Insertar valores a los parámetros de configuración de los elementos del dispositivo.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de insertar valores a los parámetros de configuración de los elementos del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Deben existir parámetros de configuración en la zona de trabajo.
Entradas/ Pasos de ejecución: • Seleccionar o escribir el valor deseado en el campo de edición del parámetro de configuración elegido.
Resultado esperado: Debe aparecer el valor insertado en el campo de edición del parámetro de configuración seleccionado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 35. Caso de prueba 6

Caso de prueba de aceptación: 6
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Modificar valores a los parámetros de configuración de los elementos del dispositivo.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de modificar valores a los parámetros de configuración de los elementos del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Deben existir parámetros de configuración en la zona de trabajo.
Entradas/ Pasos de ejecución: • Seleccionar o escribir el nuevo valor en el campo de edición del parámetro de configuración elegido.
Resultado esperado: Debe aparecer el valor modificado en el campo de edición del parámetro de configuración seleccionado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 36. Caso de prueba 7

Caso de prueba de aceptación: 7
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Insertar los valores de las variables de entrada.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de insertar los valores de las variables de entrada del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Debe estar seleccionado el elemento “Entradas” del árbol de proyecto.
Entradas/ Pasos de ejecución: • Oprimir el botón Adicionar que se encuentra en la zona de trabajo. • Seleccionar el tipo de dato de la variable de entrada. • Insertar la dirección de la variable de entrada.
Resultado esperado: Deben aparecer los valores insertados en la tabla de las variables de entrada.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 37. Caso de prueba 8

Caso de prueba de aceptación: 8
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Eliminar valores de las variables de entrada.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad que permite eliminar valores de las variables de entrada del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Deben existir valores en la tabla de las variables de entrada.
Entradas/ Pasos de ejecución: • Seleccionar la o las filas a eliminar(en caso ser más de una seleccionar presionando la tecla Ctrl) • Oprimir el botón “Eliminar” de la tabla de variables de entrada.
Resultado esperado: Deben eliminarse los valores seleccionados.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 38. Caso de prueba 9

Caso de prueba de aceptación: 9
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Modificar valores de las variables de entrada.
Responsable: Lauren San Juan Guía.
Descripción Esta prueba se realiza con el objetivo de verificar si la funcionalidad de modificar los valores de las variables de entrada del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Deben existir valores en la tabla de las variables de entrada.
Entradas/ Pasos de ejecución: • Seleccionar o escribir el nuevo valor.
Resultado esperado: Deben modificarse los valores seleccionados.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 39. Caso de prueba 10

Caso de prueba de aceptación: 10
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Insertar los valores de las variables de salida.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de insertar los valores de las variables de salida del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Debe estar seleccionado el elemento “Salidas” del árbol de proyecto.
Entradas/ Pasos de ejecución: oprimir el botón Adicionar que se encuentra en la zona de trabajo • Seleccionar el tipo de dato de la variable de salida. • Insertar la dirección de la variable de salida.
Resultado esperado: Deben aparecer los valores insertados en la tabla de las variables de salida.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 40. Caso de prueba 11

Caso de prueba de aceptación: 11
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Eliminar valores de las variables de salida.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de eliminar valores de las variables de salida del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Deben existir valores en la tabla de las variables de salida.
Entradas/ Pasos de ejecución: • Seleccionar la o las filas a eliminar(en caso ser más de una seleccionar presionando la tecla Ctrl) • Oprimir el botón “Eliminar” de la tabla de variables de salida.
Resultado esperado: Deben eliminarse los valores seleccionados.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 41. Caso de prueba 12

Caso de prueba de aceptación: 12
Historia de usuario: Gestionar los parámetros de configuración del dispositivo.
Nombre: Modificar valores de las variables de salida.
Responsable: Lauren San Juan Guía.
Descripción Esta prueba se realiza con el objetivo de verificar si la funcionalidad de modificar los valores de las variables de salida del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Deben existir valores en la tabla de las variables de entrada.
Entradas/ Pasos de ejecución: • Seleccionar o escribir el nuevo valor.
Resultado esperado: Deben modificarse los valores seleccionados.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 42. Caso de prueba 13

Caso de prueba de aceptación: 13
Historia de usuario: Salvar y cargar configuración local.
Nombre: Salvar la configuración local del dispositivo en un archivo XML.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de salvar la configuración local del dispositivo, funciona conforme a lo esperado.
Condiciones de ejecución: Debe estar abierta alguna configuración del dispositivo.
Entradas/ Pasos de ejecución: • Seleccionar del menú la opción “Salvar Proyecto”. • Ubicar en algún directorio el archivo XML creado.
Resultado esperado: Debe crearse en la ubicación del directorio seleccionada, el archivo XML con la configuración del dispositivo que fue salvado. En caso que el usuario omita la condición de ejecución, debe mostrarse el mensaje: “No existe ningún dispositivo abierto”
Evaluación de la prueba: Prueba satisfactoria.

Tabla 43. Caso de prueba 14

Caso de prueba de aceptación: 14
Historia de usuario: Salvar y cargar configuración local.
Nombre: Cargar la configuración local de un dispositivo.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad que permite cargar en el árbol de proyecto la configuración de un dispositivo procedente de un archivo local XML, funciona conforme a lo esperado.
Condiciones de ejecución: Debe existir algún archivo XML en alguna ubicación del directorio local que contenga la configuración de un dispositivo.
Entradas/ Pasos de ejecución: • Seleccionar del menú la opción “Abrir Proyecto” • Localizar el archivo XML deseado.
Resultado esperado: Debe cargarse en el árbol de proyecto la configuración del dispositivo procedente del archivo XML seleccionado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 44. Caso de prueba 15

Caso de prueba de aceptación: 15
Historia de usuario: Salvar y cargar configuración remota
Nombre: Salvar la configuración del dispositivo en el PLC-HMI.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad salvar la configuración del dispositivo en el PLC-HMI, funciona conforme a lo esperado.
Condiciones de ejecución: Debe estar abierta alguna configuración del dispositivo.
Entradas/ Pasos de ejecución: • Seleccionar la opción del menú “Descargar”. • Escribir el IP del dispositivo.
Resultado esperado: Mostrar mensaje de confirmación con el texto “Descarga exitosa”.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 45. Caso de prueba 16

Caso de prueba de aceptación: 16
Historia de usuario: Salvar y cargar configuración remota
Nombre: Cargar la configuración del dispositivo PLC-HMI.
Responsable: Lauren San Juan Guía.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad que permite cargar en el árbol de proyecto la configuración del dispositivo PLC-HMI, funciona conforme a lo esperado.
Condiciones de ejecución:
Entradas/ Pasos de ejecución: • Seleccionar la opción del menú “Cargar”. • Escribir el IP del dispositivo.
Resultado esperado: Debe cargarse en el árbol de proyecto la configuración del dispositivo PLC-HMI y mostrar el mensaje “Carga exitosa”.
Evaluación de la prueba: Prueba satisfactoria.

3.3 Ejecución de los casos de pruebas de aceptación

El proceso de pruebas a cualquier software se realiza a través de iteraciones, donde, a medida que se procede con una nueva iteración deben haberse erradicado los defectos

encontrados en la anterior, para garantizar que al final del proceso el producto quede libre de la mayor cantidad de errores posible y listo para entregar al cliente.

Durante la ejecución de los casos de pruebas de aceptación algunas no arrojaron los resultados esperados. A continuación se listan los casos de prueba de aceptación que arrojaron alguna no conformidad durante la ejecución de la primera iteración de pruebas:

1. Salvar la configuración local del dispositivo en un archivo XML.
2. Modificar valores de las variables de entrada.
3. Modificar valores de las variables de salida.

La plantilla de no conformidades constituye un registro de los defectos y fallos encontrados en el transcurso de las pruebas, cuyo principal objetivo es verificar en un futuro que estos errores fueron erradicados en posteriores iteraciones.

La siguiente tabla muestra un resumen del registro de defectos y dificultades encontradas durante la primera iteración de pruebas:

Tabla 46. Resumen de defectos y dificultades

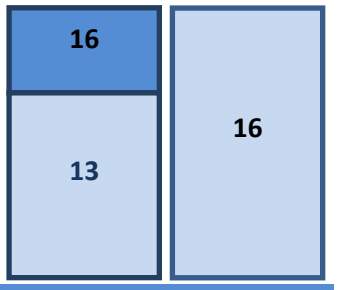
Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Respuesta del desarrollador
Interfaz de Apus-PLC	1	Se intentó salvar un proyecto que no existía y no mostró el mensaje de advertencia deseado.	Menú general en la opción "*Salvar proyecto*"	Prueba de aceptación	Resuelta una vez terminada la primera iteración de pruebas
Interfaz de Apus-PLC	2	La modificación permitió dejar el campo vacío de la dirección de la variable de entrada.	Tabla de variables de entrada campo de "*Dirección*"	Prueba de aceptación	Resuelta una vez terminada la primera iteración de pruebas
Interfaz de Apus-PLC	3	La modificación permitió dejar el campo vacío de la dirección de la variable de salida.	Tabla de variables de salida campo de "*Dirección*"	Prueba de aceptación	Resuelta una vez terminada la primera iteración de pruebas

3.4 Resultados

Para que un proceso de pruebas tenga éxito se requiere de un análisis final de los resultados arrojados, es decir, la evaluación del producto que se está probando de acuerdo a todos los defectos y fallos del sistema encontrados a lo largo del proceso.

Las pruebas de aceptación se realizaron en dos iteraciones y se utilizaron los dieciséis casos de pruebas diseñados, para verificar que las funcionalidades implementadas fueron las acordadas en las historias de usuario y que respondían correctamente a sus necesidades.

A continuación se muestra un cuadro resumen que muestra los resultados obtenidos en las pruebas ejecutadas durante las dos iteraciones realizadas:



Casos de prueba diseñados	16	16
Casos de prueba ejecutados	16	16
Casos de prueba exitosos	13	16

Figura 9. Resumen de los casos de pruebas y sus resultados

Como se muestra en la Figura 7, de los 16 casos de pruebas diseñados 13 resultaron exitosos, existiendo 3 que lanzaron una no conformidad. Por tal motivo, una vez resueltos los defectos detectados se pasó a una segunda iteración de pruebas, donde se volvieron a realizar todos los casos de prueba diseñados, para verificar que fueron resueltas las no conformidades de la primera iteración y que además no habían sufrido cambios los casos de pruebas exitosos de la primera iteración. Al concluir la segunda iteración, todos los casos de prueba arrojaron resultados satisfactorios.

Conclusiones

Después de un análisis de los resultados obtenidos, se puede concluir que el proceso de pruebas se llevó a cabo de forma satisfactoria, pues los casos de prueba desarrollados, mostraron muchos de los errores del sistema. Estos se pueden clasificar como no catastróficos, ya que se solucionaron de forma rápida y sin mayores problemas para la segunda iteración. La herramienta de configuración Apus-PLC ha quedado libre de todos los defectos encontrados en el transcurso de las pruebas y cumple eficientemente cada una de las funcionalidades acordadas con el cliente; por lo que se puede asegurar que cubre las expectativas de éste, una vez desplegado en el entorno para el cual fue construido.

Conclusiones generales

Al finalizar las etapas por la que transitó el software: investigación y selección de las tecnologías a emplear, captura de las funcionalidades a través de las historias de usuario, elaboración del diseño e implementación de la herramienta propuesta y evaluación de la herramienta construida, se arriban a las siguientes conclusiones:

- El análisis de aplicaciones homólogas a nivel mundial, demostró que estas herramientas de configuración no cumplen con las características técnicas del PLC-HMI, puesto que son específicas para cada dispositivo, aunque ayudaron a definir los elementos de configuración del hardware del PLC-HMI.

- Se obtuvo una aplicación multiplataforma desarrollada a partir del uso de tecnologías y herramientas de software libre logrando así la independencia tecnológica.

- El desarrollo de la herramienta Apus-PLC, ha facilitado el proceso de configuración del PLC-HMI, permitiendo que cuente con una herramienta informática para configurar los parámetros de ejecución del mismo y pueda ser utilizado en el ambiente industrial deseado.

Recomendaciones

Después de analizar los resultados obtenidos en el proceso de investigación y desarrollo de la herramienta de configuración, se recomienda lo siguiente:

- Integrar la herramienta Apus-PC a los módulos concebidos del PLC-HMI que son actualmente desarrollados por integrantes de la línea Sistemas Empotrados, para que sea un proceso de automatización completo.
- Desarrollar una segunda versión de la herramienta Apus-PLC que incorpore el proceso de programación del PLC-HMI.

Referencias Bibliográficas

1. **Gutierrez Ramírez, Miguel Ángel.** *Controladores Lógicos Programables.* Ingenieria Mecatrónica, Instituto Tecnológico de Celaya. Celaya : s.n., 2009. Tesis.

2. **Abarca, Patricio.** El ABC de la Automatización. Sistemas de Control Automático. [En línea] 2008. [Citado el: 15 de Febrero de 2012.] http://www.aie.cl/files/file/comites/ca/abc/sistemas-de-control-automatico.pdf.

3. **Molinari, Ing Norberto.** Curso sobre Controladores Lógicos (PLC). *EduDevices.* [En línea] 2011. [Citado el: 8 de Febrero de 2012.] Entrega No. 1. www.edudevices.com.ar.

4. **Katsuhiko, Ogata.** *Ingeniería de Control moderna.* Cuarta Edición. Madrid : Pearson Educación S.A., 2003. ISBN: 84-205-3678-4.

5. **Balcells, Josep y Romeral, José Luis.** *Autómatas programables.* s.l. : Editorial Alfaomega, 1997.

6. **Angulo Usategui, José M. y Angulo Martínez, Ignacio.** *Microcontroladores «PIC». Diseño práctico de aplicaciones.* Tercera edición. Madrid : EDIGRAFOS, S. A., 2003. ISBN: 84-481-3788-4.

7. **Molinari, Ing Norberto.** Curso sobre Controladores Lógicos (PLC). *EduDevices.* [En línea] 2010. [Citado el: 8 de Febrero de 2012.] http://www.edudevices.com.ar.

8. **Hugh, Jack .** *Automating Manufacturing Systems with PLCs.* 2008. Version 5.1, March 21, 2008.

9. **Porras Criado, Alejandro y Montero Molina, Antonio Plácido.** *Autómatas Programables.Fundamentos, manejo,instalación y practicas.* s.l. : McGraw-Hill, 1990. pág. 211. ISBN: 8476154933, 9788476154939.

10. **Siemens.** Automatización y control. [En línea] 10 de febrero de 2008. [Citado el: 15 de Febrero de 2012.] http://ingex.blogspot.com/.

11. Manual del sistema de automatización S7-200. [En línea] 2005. [Citado el: 15 de Febrero de 2012.] http://support.automation.siemens.com.

12. **Schneider.** Manual Relés programables Zelio Logic. [En línea] 2010. [Citado el: 15 de Febrero de 2012.] http://www.schneider-electronic.com.ar.

13. **Moeller.** Ayuda EASY-SOFT 6. [En línea] 2006. [Citado el: 20 de Febrero de 2012.] http://www.moeller.net.

14. **Manual y Instalador Easy Soft**. [En línea] 5 de Noviembre de 2010. [Citado el: 20 de Febrero de 2012.] http://clubelectromecanica.blogspot.com/2010/11/manual-y-instalador-easy-soft.html.

15. **Letelier, Patricio y Penadés, Mª Carmen.** Métodologías Ágiles en el Desarrollo de Software. [En línea] 2007. [Citado el: 10 de Enero de 2012.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

16. **Amaro Calderón, Sarah Dámaris y Valverde Rebaza , Jorge Carlos.** Metodologías Ágiles. [Documento]. Trujillo, Perú : s.n., 2007. Universidad Nacional de Trujillo. Facultad de Ciencias Físicas y Matemáticas.

17. **Maite Rodríguez Corbea, Meylin Ordóñez Pérez.** *La metodolgía XP aplicable al desarrollo del software educativo en Cuba.* Ciudad de la Habana : Universidad de las Ciencias Informáticas (UCI), 2007.

18. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *El Lenguaje Unificado de Modelado.* Madrid : Addison Wesley, 2001. ISBN: 9788483222706.

19. **Pérez Feria, Jordenys.** *Análisis de la arquitectura del módulo Interfaz Hombre Máquina del SCADA Nacional.* Ciudad de la Habana : s.n., 2008.

20. **Florentino, Deidry.** Historia del computador y evolución, arquitectura, lenguajes de programación y algoritmos. [En línea] Marzo de 2007. [Citado el: 15 de Febrero de 2012.] http://nectacellcomputer.jimdo.com/historia-del-computador-arquitectura-lenguajes-de-programaci%C3%B3n-y-algoritmos/.

21. **Calvo, Luis.** Lenguaje de Programación. [En línea] 22 de Abril de 2003. [Citado el: 15 de Febrero de 2012.] http://www.scribd.com/doc/Lenguaje-de-Programacion..

22. **González Abad, Israeldis.** *Diseño e Implementación del Sistema de Gestión de Información de los Recursos de la Facultad 3.* Ciudad de la Habana : s.n., 2008.

23. **Velasco Pérez, Daimeris y Vasconcelo Mir, Yuniesky Orlando.** *Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA.* La Habana : s.n., 2010.

24. **Jeffries, Let Ronald.** XPrograming. [En línea] 2011. [Citado el: 10 de Abril de 2011.] http://xprogramming.com/.

25. **Sommerville, Ian.** *Ingeniería del Software.* Septima Edición. Madrid : Pearson Educación SA., 2005. ISBN: 84-7829-074-5.

26. **Pressman, R. S.** *Ingeniería del Software. Un enfoque práctico.* Quinta Edicion. 2001. Capítulo 10.

27. **Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John.** *Design Patterns CD.* [CD] s.l. : Addison Wesley Longman, Inc., 1997. ISBN: 0-201-63498-8.

28. **Vargas Vento, Daisy Diana .** *Sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA".* La Habana : s.n., 2011.

29. Qt Reference Documentation. Introdution to Model/View Programming. *Asistente de Qt.*

30. **Architect, G. d.** Guía de usuario de Enterprise Architect6 de Abril de 2011. Obtenido de:. [En línea] 2007. [Citado el: 19 de Marzo de 2012.] http://www.sparxsystems.com.ar/index.php..

31. **Larman, Craig. UML y Patrones.** *Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HAL, 1999. ISBN: 970-1 7-0261-1.

32. **Cosín, J. D.** *Técnicas cuantitativas para la gestión en la ingeniería del software Netbiblo.* 2007.

Bibliografía

Anna C. Grimán, M. P. 2005.*Estrategia de Pruebas para Software OO que garantiza Requerimientos No Funcionales.* Departamento de Procesos y Sistemas, Universidad Simón Bolívar. Caracas, Venezuela : s.n., 2005.

Blanchette, Jasmin; Summerfield, Mark. 2006.*C++ GUI Programming.* Massachusetts : Prentice Hall, 2006. ISBN: 0-13-187249-4.

Bolton, W. 2006.*Programmable Logic Controllers programables.* Cuarta. Oxford : Elsevier Newnes, 2006. ISBN: 978-0-7506-8112-4.

Bruegge, B. D., A.*Ingeniería de Software Orientado a Objetos.* s.l. : Prentice Hall-Pearson.

Bryan, L.A. y Bryan, E. A. 1997.*Programmable Controllers. Theory and Implementation.* Segunda. Atlanta, Georgia : An Industrial Text Company Publication, 1997. ISBN: 0-944107-32-X.

Díaz, J. G.*Introducción al Proceso de Pruebas.* [ed.] Universidad de Sevilla. Sevilla : Departamento de Lenguajes y Sistemas Informáticos.

Fernández Escribano, Gerardo . 2002. Introducción a Extreme Programming. Ingeniería de Software II. 2002.

Fowler, Martinl. Presentation Mode. [En línea] [Citado el: 12 de abril de 2012.] http://www.martinfowler.com/eaaDev/PresentationModel.htm.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.*El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, S.A., 2000. ISBN: 84-7829-036-2.

Moeller. Sistemas de automatización. Autómatas programables PLC. [En línea] [Citado el: 10 de Enero de 2012.] Manual de esquemas Moeller 02/05. http://www.moeller.net/support.

Sande, Martín. 2004.*Programaciónen C++ con Qt bajo Entorno GNU/Linux.* 2004.

Sehara Driggs, Yosell Luis. 2008.*Implementación de un modelo para la configuración de un sistema SCADA.* Ciudad de la Habana : s.n., 2008.

Siemens. 2007. Herramientas para configurara y programar controladores SIMATIC. Software SIMATIC. [En línea] 2007. [Citado el: 17 de Enero de 2012.] www.siemens.com/automation/support.

Peñalver, G., Meneses, A. y García, S. 2010. Antofagasta : s.n., 2010. 1er Congreso Iberoaméricano de Ingeniería de Proyectos.