UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Título: Incorporación de la ejecución de algoritmos al módulo de adquisición del SCADA Guardián del ALBA.

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

Autor: Ernesto José Romero Ruiz

Tutor(a): Ing. Ana Silvia Tellería Martínez

Ing. Juan Carlos González Tamayo

La Habana, 1/11/2014

"Año 54 de la Revolución"

Declaración de autoría

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____	_____
Ernesto José Romero Ruiz Autor	Ing. Ana Silvia Tellería Martínez Tutora	Ing. Juan Carlos González Tamayo Tutor

Datos de contacto

Tutora: Ana Silvia Tellería Martínez

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero en Ciencias Informáticas

Correo: atelleria@uci.cu

Graduado de la UCI, con 8 años de experiencia en el tema de los sistemas SCADA.

Tutor: Juan Carlos Gonzáles Tamayo

Institución: Universidad de las Ciencias Informáticas

Título: Ingeniero en Ciencias Informáticas

Correo: jctamayo@uci.cu

Graduado de la UCI, con 5 años de experiencia en el tema de los sistemas SCADA.

Dedicatoria

Este trabajo se lo dedico a mi abuela Andrea y a mi abuelo Laureano quienes no pudieron ver a su nieto graduado de ingeniero.

A mi familia, en especial a mis padres Maritza e Israel, a mi hermano Israel, a mi sobrino Israel Ernesto, a mi novia Neyvis, en fin a toda mi familia.

Agradecimientos

Le agradezco a mi familia por los consejos, el apoyo y cariño brindado durante todo el transcurso de mi vida como estudiante, a mi novia por su apoyo incondicional.

A las amistades que nunca se alejaron y me ayudaron en mi formación de una manera u otra, y en especial a las chicas maravillosas Ailen, Anayansi, Aylin, Elizabeth, por su amistad, a Sandy, Luisma, Adrian, David, en fin a todas mis amistades.

Y a mi grupo 5505 los 'calentadores' de la facultad 5, a todos ellos gracias.

Resumen

En los procesos industriales se realiza el control automático mediante la ejecución de algoritmos embebidos en dispositivos de campo o cargados en los sistemas desplegados responsables del procesamiento de la información recibida del campo, que se encargan de, una vez obtenidas las variables involucradas, realizar los pasos definidos y llevar a cabo la automatización del dispositivo en cuestión.

El empleo de algoritmos a nivel de sistema es una característica común en las aplicaciones responsables del control automático de procesos que existen en el mercado. Esto se debe principalmente al nivel de confiabilidad que han ido alcanzando dichos sistemas en sus respectivas esferas y gracias a la flexibilidad y ventajas que brinda el uso de algoritmos en el sistema.

Con el objetivo de implementar dicha funcionalidad en el GALBA en su versión Miranda, se realizó el análisis de SCADA en busca de características relevantes, como los atributos a configurar. Lo cual permitió desarrollar una solución, cumpliendo con los requisitos planteados, contribuyendo a que el sistema GALBA sea más aplicable y competitivo.

Palabras Clave: algoritmo, control automático, SCADA

Índice

Índice de Tablas

Índice de Figuras

Introducción

El control es un conjunto de operaciones manuales o automáticas para vigilar el estado de un sistema (1) con el objetivo de mantenerlo lo más cercano posible a las mediciones deseadas. En los procesos industriales el control automático se realiza mediante la ejecución de funciones o algoritmos en los dispositivos de campo.

En la actualidad se emplean distintos algoritmos en las industrias, díganse de control o no, los cuales se diferencian según el uso o el objetivo que se quiera lograr. Por ejemplo, los algoritmos de control Proporcional Integral y Derivativo (PID del inglés Proportional Integral and Derivative), así como sus variantes PI (Proportional Integral) y PD (Proportional Derivative), los cuales según estadísticas representan más del 90% en el control de procesos industriales (2). Otro ejemplo lo constituyen los algoritmos de Control Automático de Generación (AGC del inglés Automatic Generation Control) como sus derivados Control de Carga-Frecuencia (LFC del inglés Load-Frequency Control) y Control de Turbina-Gobernador (TGC del inglés Turbine-Governor Control) empleados en la industria eléctrica para el ajuste centralizado de las consignas de potencia activa, manteniendo la frecuencia del sistema cercana a la nominal.

La informática industrial ha desempeñado un papel primordial en el control de procesos industriales, valiéndose de Sistemas de Supervisión, Control y Adquisición de Datos (SCADA acrónimo de Supervisory Control And Data Acquisition). Estos sistemas, mediante la comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.), se encargan de recolectar las variables de los procesos para luego aplicarle las transformaciones necesarias y dar una salida a través de la pantalla de un ordenador. (3)

En la actualidad los sistemas SCADA más destacados del mercado emplean la ejecución de algoritmos para realizar el control automático de los distintos procesos que manejan. Estos algoritmos son ampliamente usados en los procesos industriales y manufacturados; por ejemplo, en el diseño de sistemas de pilotos automáticos en la industria aeroespacial, en el diseño de automóviles y camiones en la industria automotriz, etc. También son esenciales en las operaciones industriales como el control de la presión, temperatura, humedad, viscosidad y flujo, entre otros (4).

Introducción

El Centro de Informática Industrial (CEDIN), perteneciente a la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), participa en el desarrollo del sistema SCADA “Guardián del ALBA” (GALBA) en su versión Miranda. Dicho sistema fue concebido para ser utilizado en las áreas productivas de Petróleos de Venezuela S.A. (PDVSA) con posible extensión de su aplicación hacia otros países de la Alianza Bolivariana de las Américas según las necesidades de la región.

El SCADA GALBA posee una arquitectura distribuida, compuesta por varios módulos: Comunicación (se encarga de la comunicación de los distintos módulos), Visualización (proporciona la interfaz visual para la interacción del usuario con el SCADA), Aplicaciones (encargado de configurar y ejecutar tareas, a las cuales se les pueden asociar scripts Python), Drivers o manejadores (realizan la traducción de los protocolos de comunicación específicos de cada dispositivo de campo hacia el protocolo genérico del SCADA), Adquisición (recolecta la información de campo utilizando los drivers y la procesa para su uso en los restantes módulos del SCADA), entre otros.

En las áreas de PDVSA y del sector eléctrico venezolano donde se aplicará el GALBA existen escenarios donde el control automático a realizar no resulta crítico y dejarlo a un operador sería costoso, las variables involucradas en dicho control no se encuentran en el mismo dispositivo de campo, resultando costosa la solución debido a que se requiere hardware adicional y de la puesta en comunicación y/o interacción entre los dispositivos involucrados, además en el sector eléctrico se necesitan ejecutar algoritmos con restricciones de tiempos significativas en su ejecución, inferiores a los tres segundos, lo cual impide que sean creados como tareas en el módulo de aplicaciones.

A partir de la situación descrita se identifica como **problema de investigación:** ¿Cómo incorporar al sistema SCADA Guardián del ALBA algoritmos con restricciones de tiempo en su ejecución?

Se define como **objeto de estudio** el sistema SCADA Guardián del ALBA, delimitando como **campo de acción** la carga y ejecución de algoritmos en el módulo de adquisición.

Se traza como **objetivo general:** Desarrollar la incorporación de ejecución de algoritmos al módulo de adquisición del SCADA Guardián del ALBA.

Para el cumplimiento del objetivo trazado se definieron las siguientes **tareas de investigación:**

- Revisión del estado del arte para identificar posibles tecnologías a utilizar y las características generales de los algoritmos aplicables a la solución.

- Estudio y caracterización del módulo de adquisición versión Miranda del SCADA Guardián del ALBA para identificar aspectos relevantes de su diseño e implementación a tener en cuenta para el desarrollo de la presente solución.

- Actualización de los requisitos funcionales y no funcionales del módulo de adquisición del SCADA Guardián del ALBA en los que impacte la incorporación de ejecución algoritmos.

- Actualización del modelo de diseño del módulo de adquisición a partir de los cambios que se realicen en sus requisitos.

- Implementación en el módulo de adquisición de los nuevos requisitos identificados.

- Diseño y ejecución de pruebas al módulo de adquisición para comprobar el cumplimiento de los requisitos y el alcance del objetivo de la solución.

En el desarrollo de la presente tesis se utilizaron los métodos de investigación científica siguientes:

Métodos Teóricos:

- Análisis - síntesis: en el análisis de la información obtenida en el proceso de investigación su división y en la generación de nuevos conceptos sobre la solución presentada.

- Inducción - deducción: en el análisis de la información referente al objeto de estudio y en el arribo a conclusiones generales que particularizaron en el campo de acción.

- Modelación: en el diseño de las clases y los algoritmos que intervinieron en la solución del problema planteado.

Métodos Empíricos:

- Consulta bibliográfica: en la elaboración del marco teórico de la investigación.

- Experimento: en la realización de pruebas unitarias donde se demostró el correcto funcionamiento del subsistema implementado en el presente trabajo.

- Entrevista: en las entrevistas a desarrolladores, asesores y arquitectos vinculados al SCADA GALBA.

Para el desarrollo de la presente tesis se definieron los siguientes tres capítulos:

CAPITULO 1: Fundamentación Teórica y Estado del Arte. En este capítulo se definen los principales conceptos del negocio para una mejor comprensión y entendimiento del mismo. Se hace referencia a las generalidades de los sistemas SCADA, enunciando sus principales características, requisitos y funcionamiento. También se brindan características de los algoritmos que se desean ejecutar desde el módulo de adquisición del SCADA GALBA. Se presentan las tecnologías y herramientas de software a utilizar en el desarrollo de la solución.

CAPITULO 2: Análisis y Diseño. El objetivo de este capítulo es brindar una visión detallada del sistema mediante la descripción de los requisitos funcionales y no funcionales que guían el desarrollo de la solución. En el mismo se definen los patrones de diseño a utilizar, además se presentan los diagramas de clases de diseño y sus relaciones.

CAPITULO 3: Implementación y Pruebas. En este último capítulo se brindan detalles del proceso de implementación, se muestran los diagramas de componentes y de despliegue. Finalmente se realizan pruebas al subsistema de adquisición resultante con el objetivo de comprobar el correcto funcionamiento y rendimiento del mismo.

Capítulo 1 Fundamentación Teórica y Estado del Arte

En el presente capítulo se realiza la fundamentación teórica y el estudio del estado del arte, abordando conceptos esenciales relacionados a los sistemas SCADA. Se dan a conocer las herramientas y metodologías que posteriormente serán utilizadas en el desarrollo de la tesis.

1.1 Principales Conceptos

A continuación se describen los conceptos principales a tratar en el desarrollo del presente trabajo.

Control Automático

Se ocupa del control de un proceso en un estado determinado, comparando el valor efectivo de la salida de una planta con el valor deseado, determina la desviación y produce una señal de control que reduce la desviación a cero o a un valor pequeño; por ejemplo, los dispositivos de campo empleados en la industria del petróleo, regulan la presión, temperatura, humedad, entre otros parámetros, en procesos como los de extracción, refinación, almacenamiento, etc. con la meta de acercarlo a un valor configurado. La forma en que el control automático produce la señal de control recibe el nombre de acción de control. (5)

Algoritmo

Es un conjunto finito y ordenado de instrucciones, pasos o procesos que llevan a la solución de un determinado problema. (6)

En la actualidad existen algoritmos para el control y supervisión de procesos productivos, los cuales varían con respecto a la cantidad de entradas y salidas así como de la función que cumplen cada uno de ellos, de ahí el uso en los sectores industriales de algoritmos y variantes de los mismos para el control automático de procesos. Ejemplo de ellos son la familia de algoritmos AGC, entre otros que son empleados en las industria eléctrica para la regulación de potencia generada, los algoritmos de PID y sus variantes PI y PD que tienen un papel de vital importancia en el autocontrol de dispositivos de campo, logrando

que el valor obtenido presente el menor error y ajustándolo al valor establecido por configuración.

Algoritmo PID

El algoritmo de cálculo del control PID se divide en tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional determina la reacción del error actual. El Integral genera una corrección proporcional a la integral del error, esto asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. El Derivativo determina la reacción del tiempo en el que el error se produce. (2)

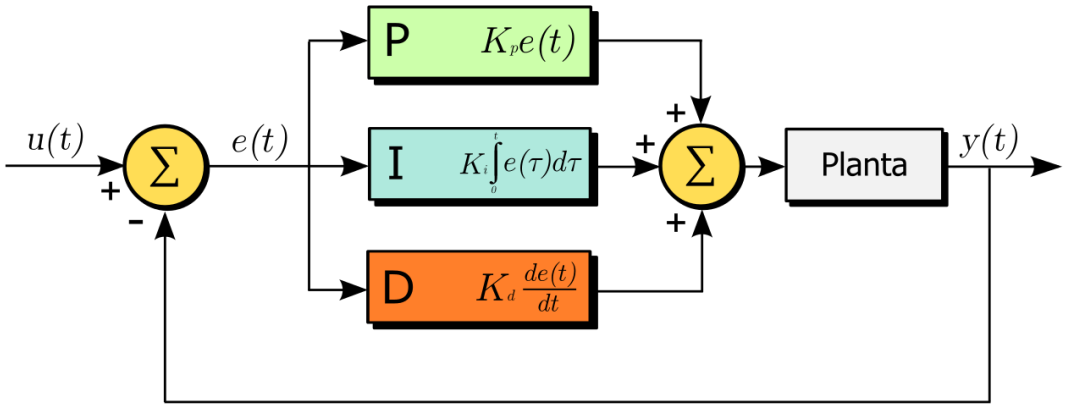


Figura 1 Diagrama de un algoritmo PID.

Algoritmo AGC

El algoritmo de cálculo del AGC se encarga del cálculo de la variación neta de generación (generalmente conocida como ACE del inglés Area Control Error) requerida, con el fin de mantener el tiempo promedio del ACE en un bajo valor. El ACE se define como una combinación lineal de las desviaciones de potencias netas de intercambio y de frecuencias, las cuales se toman generalmente como salidas del AGC. A medida que el ACE es impulsado a cero por el AGC, tanto en frecuencia como errores de la línea eléctrica se verán obligados a ceros. Los AGC pueden ser vistos como funciones de control de supervisión que intentan hacer coincidir la tendencia de generación dentro de

un área a la tendencia de cambio de carga de la zona, así como mantener la frecuencia del sistema y el flujo de energía cerca al valor establecido. (7)

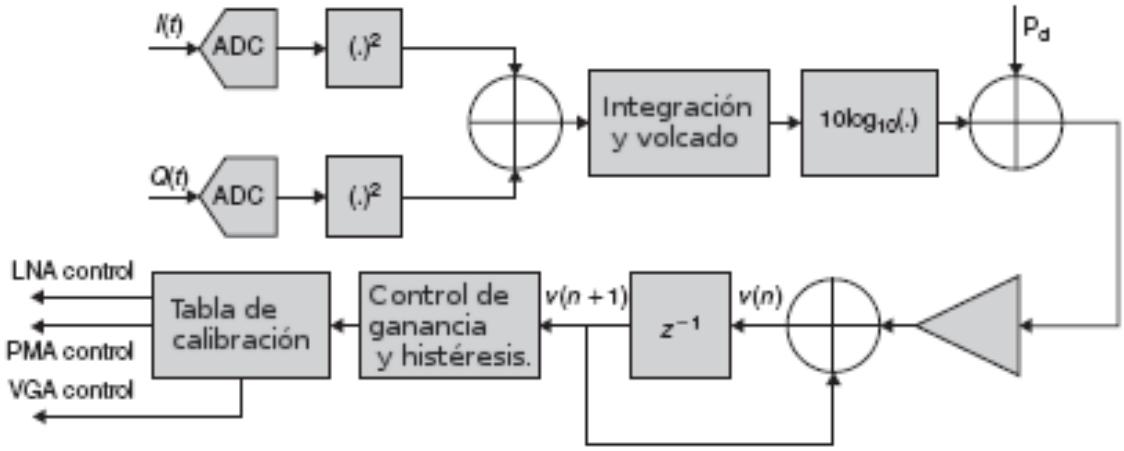


Figura 2 Diagrama de un algoritmo AGC.

Procesamiento

Es la aplicación sistemática de una serie de operaciones sobre un conjunto de datos, generalmente por medio de máquinas, para explotar la información que estos datos representan. (8)

En los sistemas SCADA el procesamiento se realiza sobre los puntos, alarmas y comandos que se obtienen o generan durante la ejecución del sistema.

1.2 Sistemas SCADA

“Basándonos en la universal *Ley del mínimo esfuerzo*, podríamos enfocar los logros tecnológicos como la consecuencia de no querer cansarnos más de lo necesario. En el caso de la informática, su nacimiento y evolución se deberían a la necesidad de querer automatizar el cálculo matemático y a no querer contar con los dedos”. (9)

Los sistemas SCADA comprenden todas aquellas soluciones de aplicación que requieren de la captura, procesamiento y visualización de la información de un proceso o planta industrial. Esta información procesada es utilizada para realizar una serie de análisis o estudios con los que se pueden obtener valiosos indicadores que contribuyan a la toma de decisiones en el control de la planta por parte de los operadores. (10)

Los SCADA mejoran la eficacia y eficiencia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas (11), al contar con la información (alarmas, históricos, entre otros) de primera mano de lo que ocurre u ocurrió en el proceso. También puede integrarse a las intranets para la visualización de la información vía web ya sea mediante un dispositivo móvil o un ordenador, sin importar el sistema operativo. Lo que lo convierte en parte de la estructura de gerenciamiento de la información corporativa. Estos sistemas ya no son vistos simplemente como herramientas operacionales, sino como un recurso importante de recopilación de información. Los datos que se pueden obtener de las estadísticas del proceso, eventos de detección de fugas, representaciones de gráficos de producción entre otros, se consideran de vital importancia para la toma de decisiones económicas. (12)

1.3 Sistemas SCADA existentes

Hoy en día existen varios sistemas SCADA que destacan por su flexibilidad a la hora de adaptarse a una solución determinada, automatizando una amplia gama de procesos industriales. Entre ellos se pueden mencionar a Simantic WinCC (Windows Control Center), el Progea Movicon, el InTouch HMI/SCADA y el OpenSCADA.

Simantic WinCC

Con una estructura modular, el sistema base WinCC combina la producción y la automatización de procesos en un solo paquete de software HMI. Además cuenta con sistemas redundantes distribuidos para las plantas, bases de datos de alto rendimiento basadas en el Microsoft SQL Server que forman el eje central de información, interfaces abiertas y capacidad de expansión flexible. (13)

Progea Movicon

Progea Movicon (del inglés Monitoring, Vision, and Control) es desarrollado por la compañía Progea, cuenta con un paquete de software diseñado para la creación de una interfaz hombre-máquina y una estación de supervisión, control y adquisición de datos (SCADA), basado en un ordenador personal. Los dispositivos de gestión de procesos, tales como PLC, controladores de temperatura, tarjetas inteligentes, PC, pueden ser

conectados al sistema en el que está instalado Movicon a través de los puertos serie, módems, redes de comunicación, etc. (14)

InTouch HMI/SCADA

El InTouch HMI/SCADA, puede crear aplicaciones con varias funciones. InTouch se compone de tres grandes programas, el Administrador de InTouch Aplicación, WindowMaker y WindowViewer. Entre las características del InTouch se encuentran un explorador de aplicaciones, control de fallo de instrumento, sistema distribuido de alarmas, visualización de proceso de aplicación en tiempo real, funciones asíncronas, históricos distribuidos, entre otras. (15)

OpenSCADA

OpenSCADA es un sistema SCADA de código abierto desarrollado bajo la licencia GPL v2 construido en los principios de modularidad y escalabilidad. El sistema OpenSCADA está dirigido a la: adquisición, visualización de la información y también para otras operaciones relacionadas. OpenSCADA se puede utilizar en instalaciones industriales; en los dispositivos integrados, como un entorno de ejecución, incluso dentro de un PLC; en los centros de datos u otras instalaciones de servidor para recopilar, procesar, presentar y archivar datos referentes a su red y entorno de ejecución. (16)

1.4 Análisis de los SCADA

Los sistemas Simantic WinCC, Progea Movicon e InTouch HMI/SCADA son privativos. Al no ser software libre sus códigos fuentes no pueden ser adquiridos para su estudio y análisis, en busca de conocimiento previo como base de desarrollo de las características deseadas en el subsistema de algoritmos. No siendo así con el OpenSCADA, permitiendo la adquisición del sistema y del código fuente de forma gratuita.

Tabla 1 Estado del arte de los SCADA.

SCADA / Funcionalidades	Simantic WinCC	Progea Movicon	InTouch	OpenSCADA

Implementación de funciones	**X**	**X**	**X**	**X**
Carga y ejecución de funciones	**X**	**X**	**X**	**X**
Control de tiempo de ejecución de las funciones	**X**	**X**	**X**	**X**

Leyenda:
Soporta: **X**

Estos SCADA tanto privativos como libres presentan funcionalidades que se asemejan a las deseadas en el GALBA, dígase las interfaces para el diseño y desarrollo de funciones (ambas estudiadas para determinar los principales atributos a configurar) así como la carga y ejecución de las mismas. Seleccionándose el OpenSCADA, con la meta de estudiar el código en busca de características relevantes para la implementación de las funcionalidades mencionadas.

1.5 Arquitectura del SCADA GALBA

El sistema Guardián del ALBA está dividido, de forma general, en diez subsistemas que permiten la escalabilidad de las soluciones. La figura siguiente, representa la interrelación de alto nivel de los subsistemas (10).

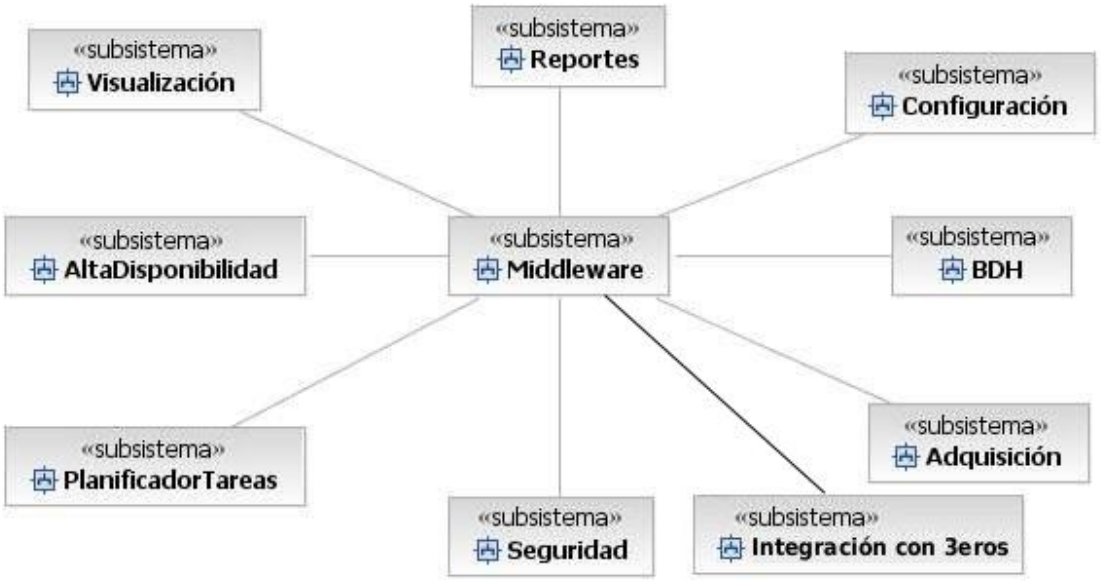


Figura 3 Diagrama de interacción de alto nivel de los subsistemas del Guardián del ALBA.

El subsistema de Middleware

Capa de comunicaciones que ofrece un conjunto de servicios a los módulos del SCADA para el envío asíncrono de la información relacionada con puntos, alarmas y comandos. Para ello hace empleo del paradigma publicación/suscripción haciendo posible el funcionamiento del SCADA en una arquitecta distribuida. (17)

El subsistema de Visualización

El módulo de Visualización o HMI (HMI del inglés Human-Machine Interface) en el SCADA se encarga de representar en un ordenador los procesos que ocurren en el campo en tiempo real, muestra los componentes implicados, los sensores, las estaciones remotas y el sistema de comunicación dándole al operador total control. Éste módulo es el que permite al operador estar en contacto directo con el sistema y realizar la supervisión y el control del proceso en general. (10)

El subsistema de Adquisición

Es el responsable de recolectar y procesar la información de los distintos dispositivos de campo para su utilización por los restantes módulos del SCADA, así como la gestión de alarmas.

Está compuesto por los módulos siguientes:

- **AcquisitionCommons:** Es una biblioteca que contiene las estructuras donde se almacena la información de los recursos de adquisición para su utilización por los restantes módulos del subsistema de adquisición. Así como las estructuras y clases para la gestión de la memoria compartida.

- **AcquisitionMonitor:** Es el proceso encargado de supervisar porque los procesos de adquisición estén corriendo correctamente, en caso de que alguno deje de funcionar lo ejecuta nuevamente. Vigila el consumo de memoria del módulo y su rendimiento.

- **AcquisitionAgentConfig:** Es el proceso responsable de cargar la configuración desde la base de datos de Configuración al iniciar el módulo. Crea e inicializa las estructuras necesarias para representar cada punto, alarma o dispositivo que haya sido configurado por el operador.

- **AcquisitionConfiguration:** Es una biblioteca que posibilita la creación de las memorias compartidas de puntos, alarmas, dispositivos y configuración, ofreciendo interfaces para el manejo de las mismas. La memoria compartida de configuración es utilizada por los demás subsistemas para hacer persistir los datos recolectados que puedan ser necesarios en un futuro procesamiento.

- **Collector:** Es el proceso que se encarga de la comunicación y procesamiento de las respuestas recibidas por los dispositivos de campos mediante los manejadores. Además crea y planifica los bloques de variables a ser encuestados.

- **ProcessingPoint:** Proceso responsable de recibir y procesar los datos provenientes del **Collector**. Entre sus principales funciones está la conversión lineal entre escalas de los datos recolectados, el procesamiento de variables calculadas, la actualización de la memoria compartida de puntos, entre otros.

- **CommandProcess:** Es el proceso que propaga a los niveles inferiores del SCADA los comandos enviados por los operadores o sistemas automatizados de control de procesos.

- **AlarmProcessing:** Es el proceso encargado de analizar cada una de las muestras recolectadas para detectar alarmas configuradas. Gestiona el número de ocurrencias de cada alarma, banda muerta a la activación, entre otros.

- **PointsPublisher:** Proceso responsable de publicar a los clientes la actualización de los puntos, alarmas y eventos.

La siguiente figura muestra un posible despliegue de los componentes que integran el SCADA GALBA:

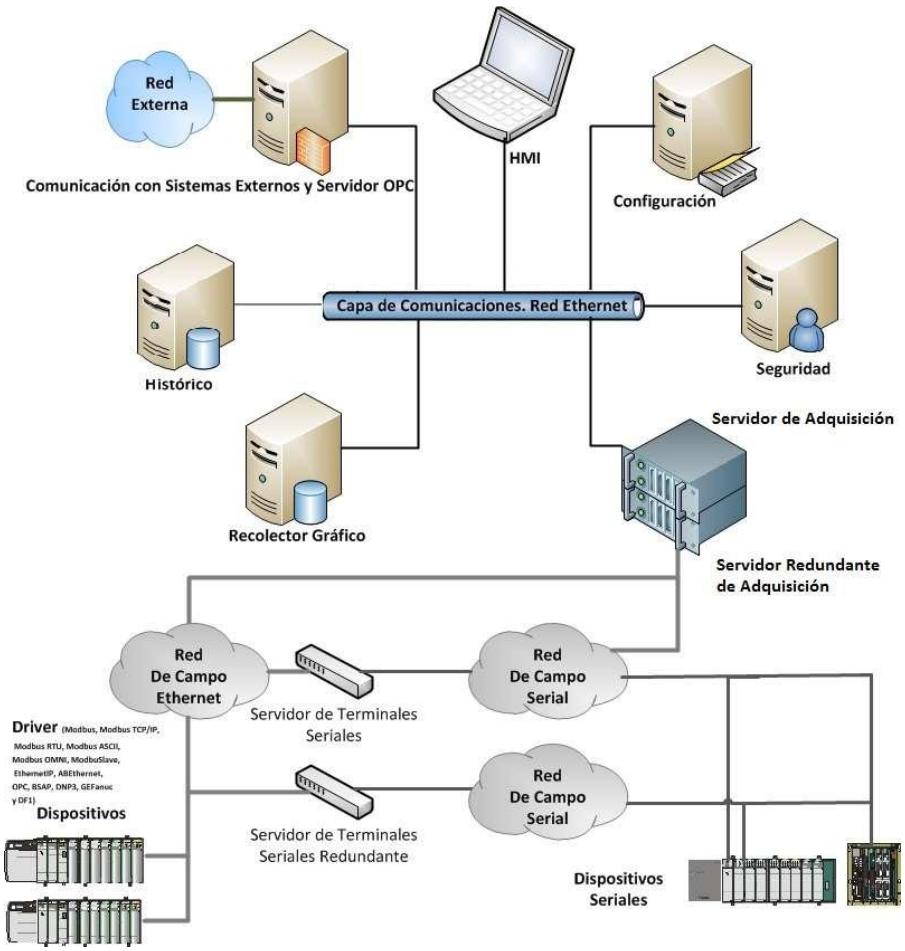


Figura 4 Despliegue de los diferentes módulos que componen el SCADA GALBA.

1.6 Tecnologías y herramientas a utilizar

A continuación se dan a conocer las tecnologías y herramientas a utilizar en el desarrollo del trabajo, los cuales ya fueron analizadas, evaluadas y aprobadas en el inicio del proyecto SCADA GALBA.

Metodología de desarrollo

Se utiliza la metodología Proceso Unificado de Rational (RUP) desarrollado por la empresa Rational Software de IBM. Sus características más relevantes son: el enfoque dirigido por casos de uso, estar centrado en la arquitectura y ser iterativo e incremental. El mismo consta de cuatro fases (Inicio, Elaboración, Construcción y Transición) en las cuales se realizan varias iteraciones con el objetivo de obtener un software con una alta calidad. (18)

Lenguaje de modelado

Se empleó el Lenguaje Unificado de Modelado (UML del inglés Unified Modeling Language), lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Prescribiendo un conjunto de notaciones y diagramas estándares para modelar sistemas orientados a objetos, describiendo la semántica esencial de lo que estos diagramas y símbolos significan. (19)

Herramientas de modelado

Se hizo uso de la herramienta de Ingeniería de Software Asistida por Computadora (CASE del inglés Computer-Assisted Software Engineering) Visual Paradigm desarrollado por la empresa Visual Paradigm International, que utiliza UML como lenguaje de modelado. Permite soportar el ciclo de vida completo del software: análisis, diseño, implementación y despliegue; así como dibujar los diagramas de clases, generar los diagramas a partir del código, generar código desde diagramas y generar documentación.

Lenguaje de programación

Se utilizó como lenguaje de desarrollo el C++, derivado de C. El mismo brinda las potencialidades del paradigma de programación orientado a objetos como abstracción, encapsulación, herencia y polimorfismo. (20)

Herramienta de Documentación

Para la generación de documentación se trabajó con el Doxygen, herramienta que permite la generación de documentación de código fuente escrito en C++, aunque también es compatible con otros lenguajes de programación como C, Objective-C, C #, PHP, Java, Python, entre otros. Entre sus características se encuentran la generación de un navegador de documentación en línea (en HTML), así como la generación de gráficos de dependencias, diagramas de herencias y de colaboración, entre otras.

Herramienta de Control de Versiones

La herramienta Subversion, publicado bajo una licencia de tipo Apache/BSD. Entre sus características se encuentran el acceso al repositorio a través de redes, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración y el seguimiento de los archivos y directorios a través de copias y renombrados, entre otras.

Entorno de desarrollo integrado (IDE)

Un IDE (del inglés Integrated Development Environment) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Se desarrollo en el IDE Eclipse, el cual emplea módulos (plugins) para la extensión de sus funcionalidades, gracias a los mencionados plugins es posible utilizar múltiples lenguajes, entre ellos C++, permite la integración con varios frameworks de desarrollo como es el caso de Qt, así como el empleo de mecanismos de control de versiones como CVS y subversion (SVN).

Capítulo 2 Análisis y Diseño

En este capítulo se expone el Análisis y Diseño realizado para modelar y dar respuesta al problema planteado. Para la representación de los conceptos, sus atributos y asociaciones se escogió realizar un Modelo de Dominio, para ganar en comprensión de la solución. Se aborda el proceso de captura de los requisitos funcionales y no funcionales de la solución propuesta.

2.1 Modelo de Dominio

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. Es una representación de las clases conceptuales del mundo real, no de componentes de software. Este tipo de modelo tiene como objetivo principal ayudar a comprender los conceptos fundamentales de subsistema de algoritmos, y a ganar en claridad acerca del funcionamiento. (21)

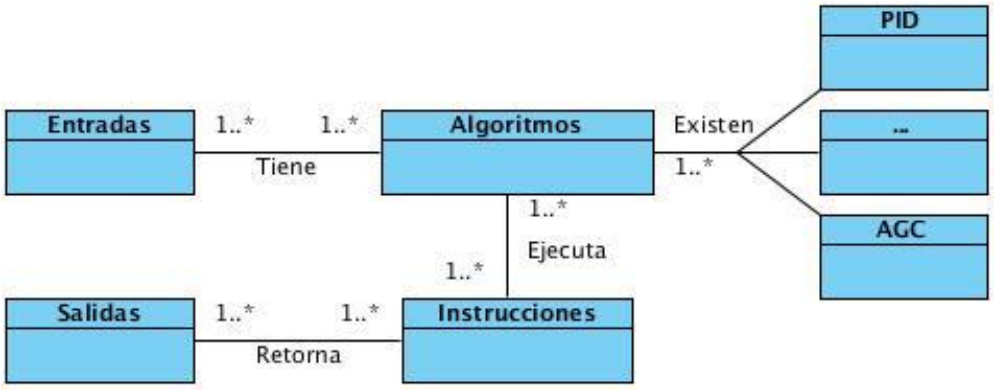


Figura 5 Modelo de dominio.

Los **algoritmos** a cargar y ejecutar se le especifican distintos parámetros de configuración para la posterior ejecución de **instrucciones**. Entre los parámetros de configuración se encuentran las **entradas**, los cuales pueden ser variables, constantes y/o valores asociados a las variables del sistema GALBA. Además se les configuran una o más **salidas**, que son asociadas a las variables del GALBA, díganse variables relacionadas a dispositivos o variables virtuales con el objetivo de obtener información relevante.

2.2 Especificaciones de requisitos de software

A continuación se enumeran los requisitos funcionales y no funcionales que recogen formalmente las características del subsistema de algoritmos.

Requisitos funcionales

RF1. Cargar configuración.

RF2. Cargar algoritmos.

RF3. Ejecutar algoritmos.

RF4. Control del tiempo de ejecución de los algoritmos.

RF5. Escribir resultado de algoritmos.

Requisitos no funcionales

RFN1. Uso de tecnologías libres: El sistema debe ser implementado con tecnologías libres.

RFN2. Empleo de las herramientas definidas para el desarrollo del GALBA: Eclipse, Doxygen, Subversion, etc.

RFN3. Codificación según el estilo de código definido en el documento ESCMGA-SGA-0120_1_EstandarC++ (22).

Descripción de los requisitos funcionales

A continuación se describirán los requisitos funcionales identificados para el desarrollo de la solución.

Tabla 2 Requisitos funcionales.

No	Nombre	Descripción	Complejidad
RF1	Cargar configuración.	Leer la configuración recibida del módulo de configuración, la cual está	Media

		compuesta por una serie de datos referentes a los algoritmos a ejecutar, díganse las entradas, salidas donde se escribirá el resultado la ejecución, variables o constantes configuradas por el operador del sistema necesarias para el correcto funcionamiento del algoritmo, así como la dirección de donde se encuentra el algoritmo, el nombre, además del número de reintentos en caso de falla durante la ejecución, periodo de en qué se ejecutara el algoritmo, y el tiempo que se estima dure su ejecución.	
RF2	Cargar algoritmos.	Los algoritmos especificados previamente en el RF1, se les realizara el proceso de carga, durante el cual se realiza la comprobación de que los datos referentes al algoritmo están correcto, en caso de presentar errores se creara un log describiendo la situación ocurrida, si durante el proceso de carga no ocurrieron errores se crea un log indicando que el algoritmo está listo para su ejecución. Al finalizar se crea un log informando de cuantos fueron configurados correctamente y cuantos no. Los algoritmos a cargar pueden variar en dependencia de la cantidad de entradas y salidas de los mismos, por ejemplo que tengan una entrada y varias salidas o varias entradas y varias salidas, o cualquier	Alta

		combinación.	
RF3	Ejecutar algoritmos.	Para la ejecución de los algoritmos primero se realizara una captura de todas entradas relacionadas al SCADA creando una instantánea para prevenir cambios de las mismas durante la ejecución, la misma se le pasara al algoritmo para la actualización de los valores correspondientes y luego se procederá con la ejecución, una vez terminada se realiza la lectura de las salidas.	Alta
RF4	Control del tiempo de ejecución de los algoritmos.	Se vela por que los algoritmos se ejecuten en el tiempo estimado especificado previamente en el RF1, de no ser así se interrumpe la ejecución del mismo, se reconfigura e inicializa nuevamente dejándolo listo para su próxima ejecución, se incrementa el contador de reintentos fallidos y se crea un log informando de lo sucedido.	Alta
RF5	Escribir resultado de algoritmos.	Una vez realizada la ejecución y lectura de las salidas del algoritmo, se actualizan los puntos previamente configurados como salidas de los algoritmos.	Media

2.3 Modelo de Diseño

En lo adelante se expondrán los componentes del subsistema de algoritmos, así como la estructura de sus clases y su relación con los subsistemas que interactúa para el cumplimiento de los requisitos funcionales planteados anteriormente. Se darán a conocer los principales métodos de las clases correspondientes y sus funcionamientos en dicho subsistema.

2.3.1 Diagrama de Paquetes

Se identificaron cuatro paquetes con los cuales interactúa el subsistema de algoritmos. Las dependencias y relaciones entre estos paquetes se muestran en la figura que sigue.

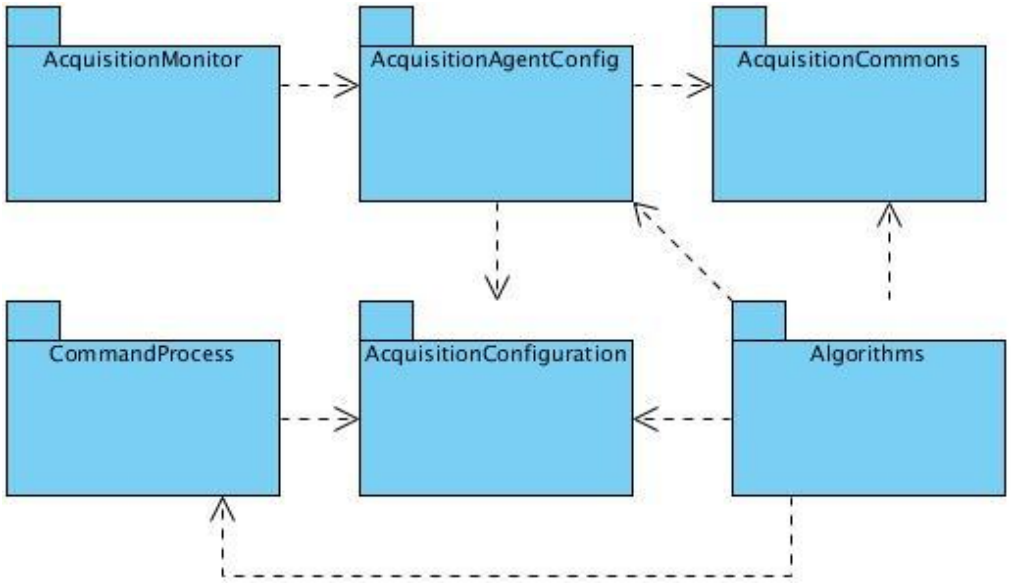


Figura 6 Diagrama de paquetes del subsistema de algoritmos.

El subsistema de **AcquisitionMonitor** es el encargado de la inicialización del módulo de adquisición. Una vez terminado notifica a **AcquisitionAgentConfig** el cuál es el responsable de cargar la configuración de los distintos módulos y ponerla a disposición de los restantes módulos mediante la memoria compartida. Posteriormente el acceso a la memoria compartida de configuración se realizará mediante el **AcquisitionConfiguration**. Ya terminado de cargar la configuración el **AcquisitionMonitor** notifica a los subsistemas que pueden iniciar la ejecución.

El subsistema **Algorithms** una vez que recibe la notificación del **AcquisitionMonitor** realiza el proceso de configuración y posterior ejecución de los algoritmos.

2.3.2 Diagrama de Clases

Se definieron ocho clases y una estructura para el desarrollo de la solución propuesta. En la siguiente figura se muestra el diagrama de clases correspondiente.

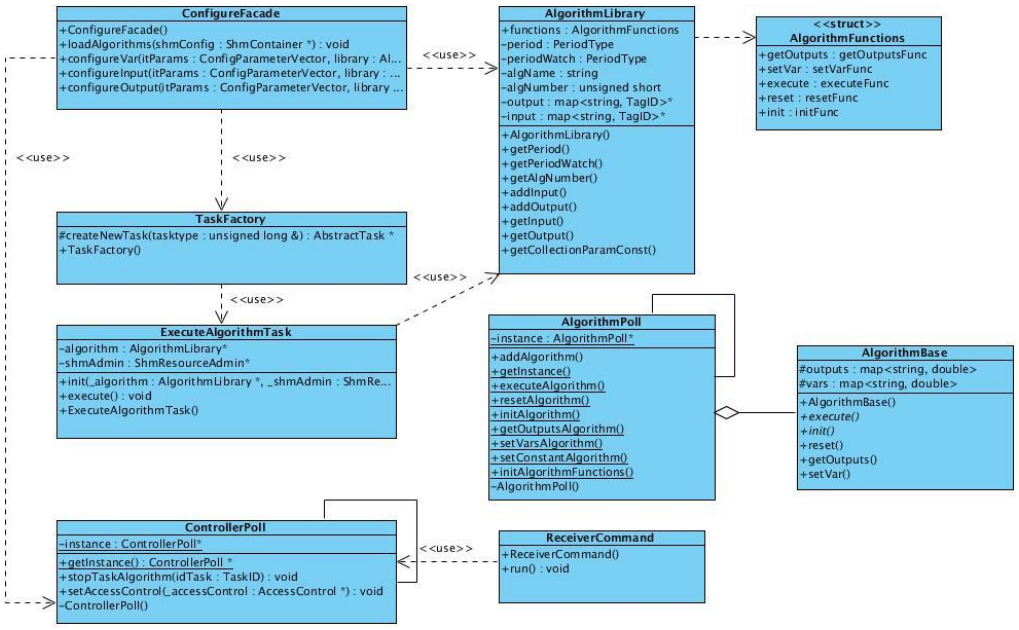


Figura 7 Diagrama de clases del subsistema de algoritmos.

2.3.3 Patrones de Diseño

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software”. Un patrón de diseño puede ser caracterizado como "una regla de tres partes que expresa una relación entre un cierto contexto, un problema, y una solución". Para el diseño de software, el contexto permite al lector comprender el entorno en el que el problema reside y qué solución podría ser apropiada dentro de ese entorno una serie de requisitos, incluidas las limitaciones y restricciones, actúa como un sistema de fuerzas que influyen en cómo el problema se puede interpretar dentro de su contexto y cómo la solución puede ser aplicado efectivamente (23). En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

A continuación se describen los empleados en la implementación de la solución:

Instancia Única

Fue utilizado en la clase AlgorithmPoll y ControllerPoll para garantizar que la clase sólo tenga una única instancia y proporcionar un punto de acceso global a ésta instancia.

A continuación se muestra el diagrama formal del patrón y una representación de la clase donde se empleó.

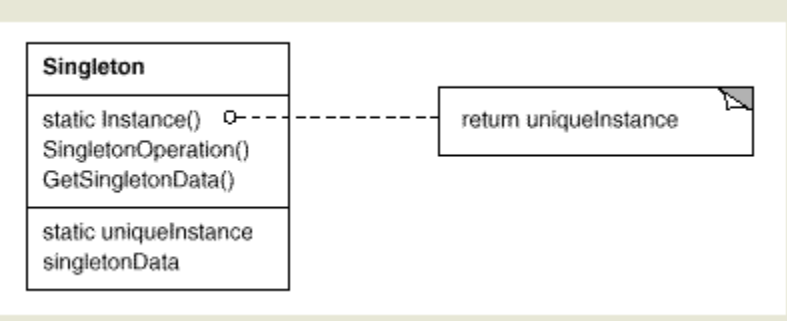


Figura 8 Diagrama del patrón Instancia Única.

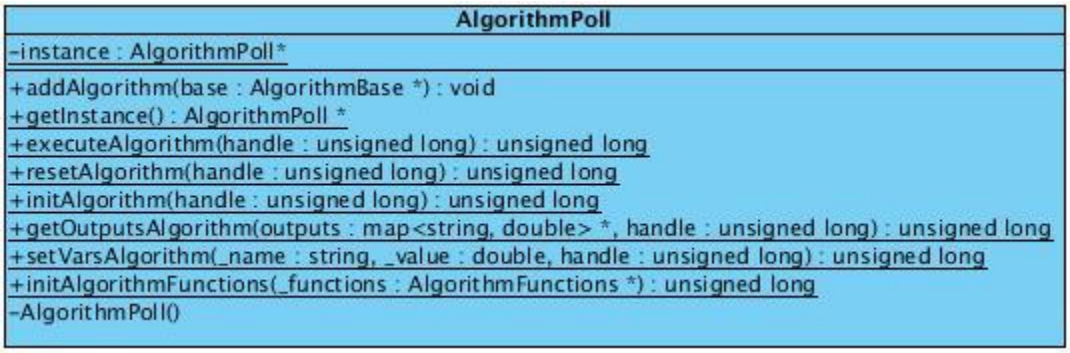


Figura 9 Clase donde se utilizó el patrón Instancia Única.

Método de Fábrica

Se usó en la definición de una interfaz para crear objetos mediante uno de sus métodos, en nuestro caso particular se empleó en la creación de las tareas que se encargan de la ejecución de los algoritmos.

A continuación se muestra el diagrama formal del patrón y una representación de las clases donde se utilizó.

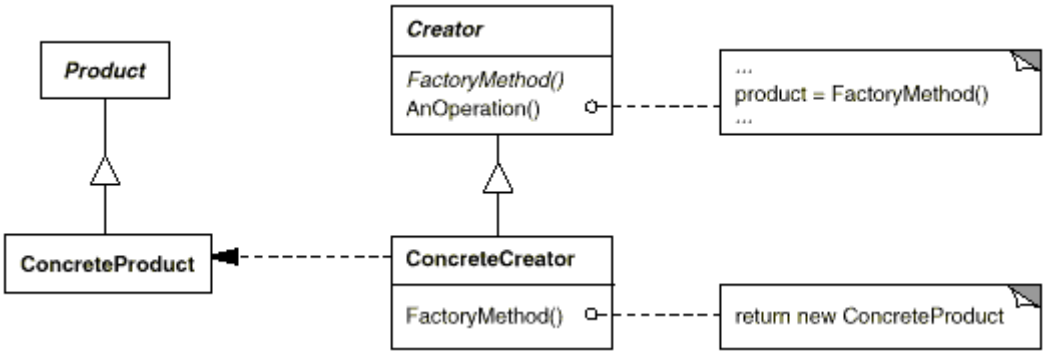


Figura 10 Diagrama del patrón Método de Fábrica.

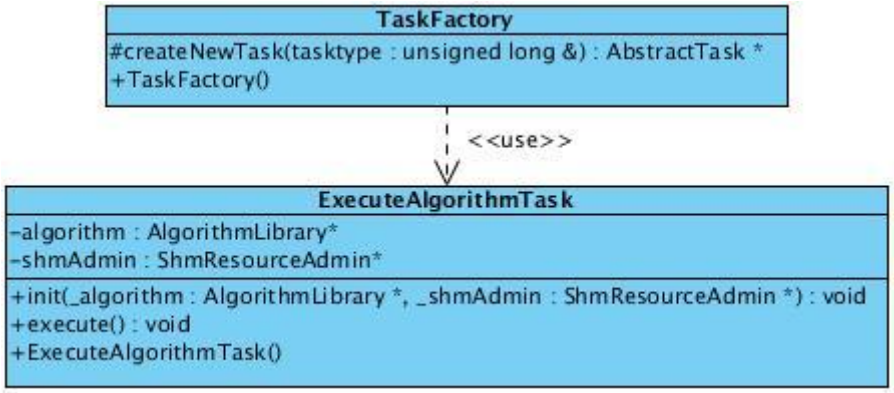


Figura 11 Clase donde se utilizó el patrón Método Fábrica.

Estrategia

Se utilizó para definir la familia de algoritmos encapsulando por separado cada uno de ellos y haciéndolos, por tanto, intercambiables.

A continuación se muestra el diagrama formal del patrón y una representación de la clase donde se empleó.

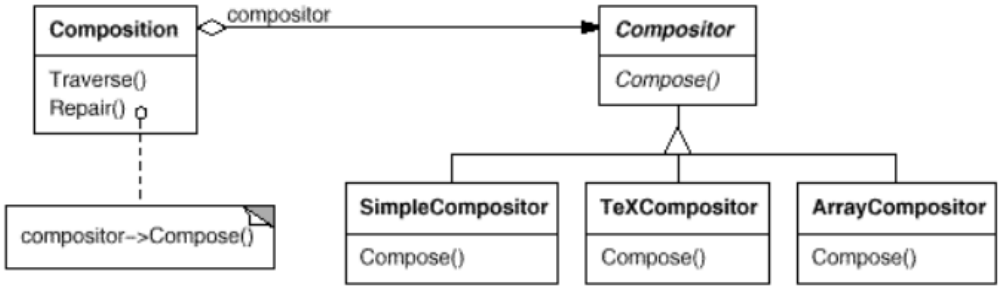


Figura 12 Diagrama del patrón Estrategia.

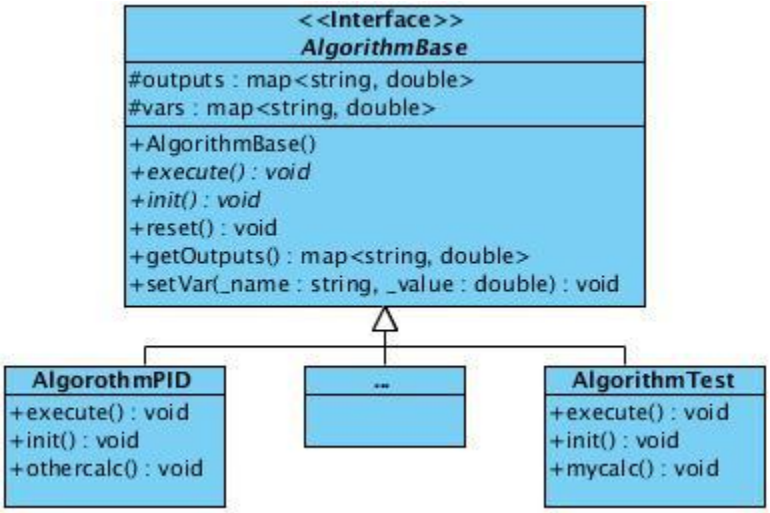


Figura 13 Clase donde se utilizó el patrón Estrategia.

Descripción del diagrama de clases del subsistema de algoritmos

A continuación se realizara una descripción referente a las clases y estructura expuestas en la figura anterior para una mayor comprensión.

Clase ConfigureFacade

Es la encargada de la carga de la configuración disponible en la memoria compartida, así como la responsable de la inicialización de los objetos correspondientes para el trabajo del subsistema. La misma configura los parámetros necesarios para la correcta ejecución de los algoritmos cargados en memoria para su posterior ejecución.

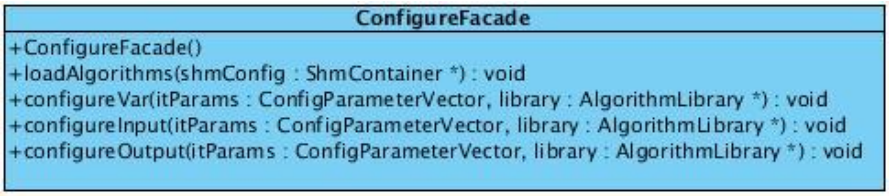


Figura 14 Diagrama de Clase ConfigureFacade.

Clase AlgorithmBase

Es la interfaz que se brinda para el posterior desarrollo de los algoritmos a cargar por el subsistema. Dicha clase está compuesta por los elementos y/o atributos fundamentales que se necesitan para la declaración y definición de algoritmos, entre los atributos se encuentran inputs y outputs, mapas donde se guardan las entradas y las salidas respectivamente del algoritmo, así como los métodos para el trabajo con los atributos de la clase.

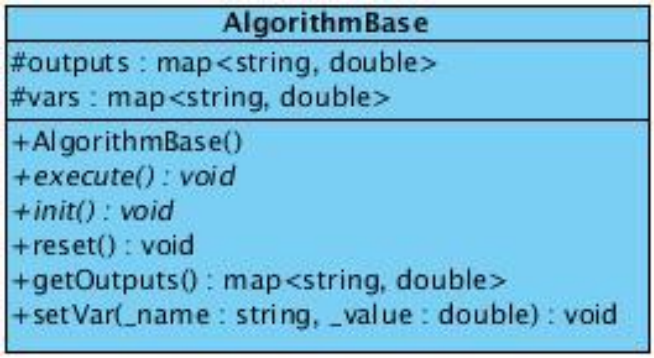


Figura 15 Diagrama de Clase AlgorithmBase.

Clase AlgorithmLibrary

Es la representación de las bibliotecas que contienen los algoritmos y sus atributos, dentro de los cuales se encuentran el período de ejecución, el tiempo en que se deben ejecutar los algoritmos configurados, además se encuentra la estructura que contiene los punteros a funciones correspondientes para el trabajo con la biblioteca mediante los cuales se configuran y ejecutan los algoritmos.

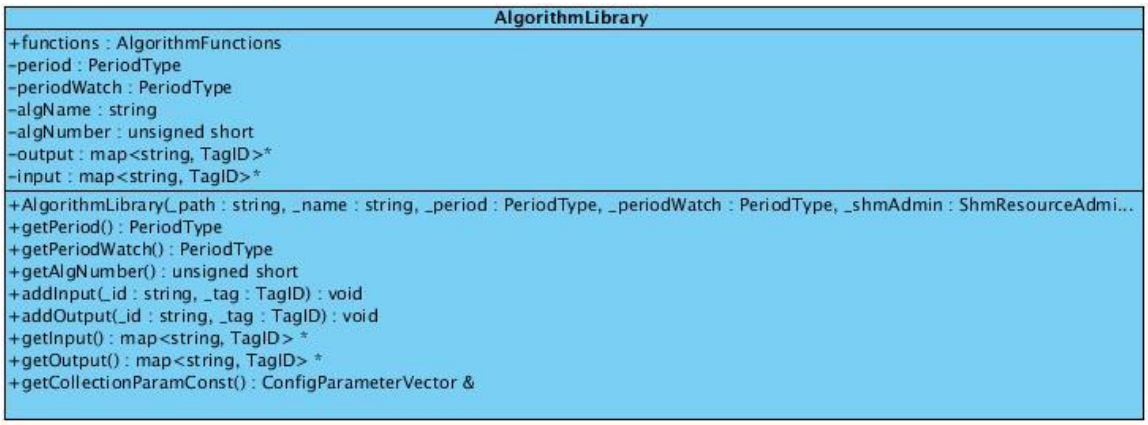


Figura 16 Diagrama de Clase AlgorithmLibrary.

Clase AlgorithmPoll

Se encarga de mantener en memoria los algoritmos cargados desde las bibliotecas, a través sus métodos se realiza la ejecución de los diferentes métodos contenidos en los algoritmos, entre los cuales están la configuración de las entradas, las salidas y las constantes, la obtención de las salidas, y el método initAlgorithmFunctions que es el encargado de inicializar los punteros a funciones contenidos en uno de los atributos de la clase **AlgorithmLibrary**.

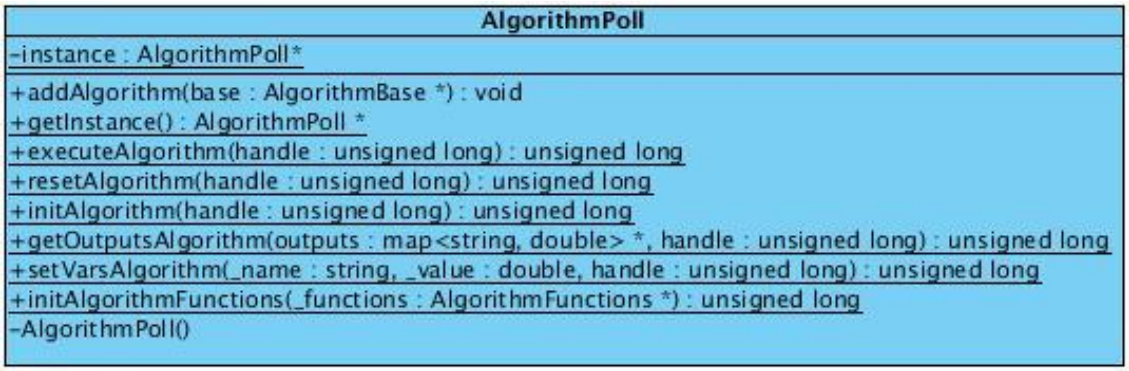


Figura 17 Diagrama de Clase AlgorithmPoll.

Estructura AlgorithmFunctions

Es donde se encuentran los punteros a funciones, los cuales son empleados en la configuración de los algoritmos, así como en su ejecución y reinicio en caso de problemas durante la ejecución.

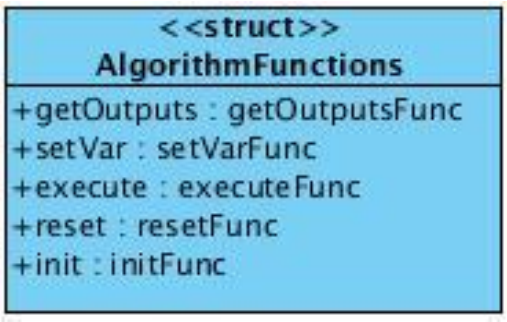


Figura 18 Diagrama de estructura AlgorithmFunction.

Capítulo 3 Implementación y Pruebas

En este capítulo se muestran los flujos de trabajo de implementación y pruebas. Se realizan pruebas a los requisitos funcionales obtenidos y se valida el correcto funcionamiento de los mismos, verificando una correcta integración de la solución. Se realiza una valoración del resultado obtenido.

3.1 Implementación

En la fase de construcción de un software es donde se realiza la codificación de las clases y los objetos en archivos fuentes y ejecutables, obteniéndose como resultado una aplicación que recoge las funcionalidades definidas en la captura de requisitos.

3.1.1 Diagrama de Componentes

Se empleó para ilustrar las piezas del software que conformarán el subsistema, ya que posibilita un nivel más alto de abstracción que un diagrama de clases.

A continuación se muestra el diagrama de componentes de la solución y sus relaciones.

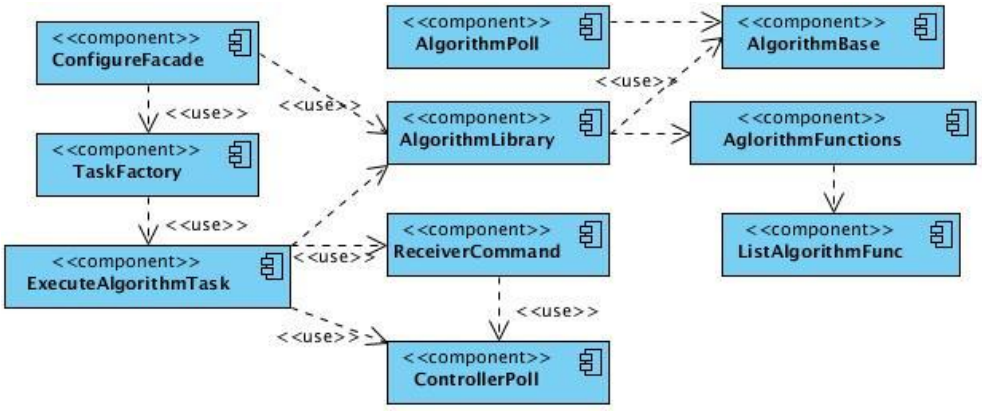


Figura 19 Diagrama de componentes del subsistema de algoritmos.

3.1.2 Métodos relevantes

A continuidad se describen dos de los métodos empleados para dar cumplimiento a las funcionalidades de la solución.

Cargar Configuración

Una vez recibido el mensaje de que se puede iniciar el subsistema, este comienza el proceso de cargar la configuración, el cual se describe a continuación:

Pseudocódigo

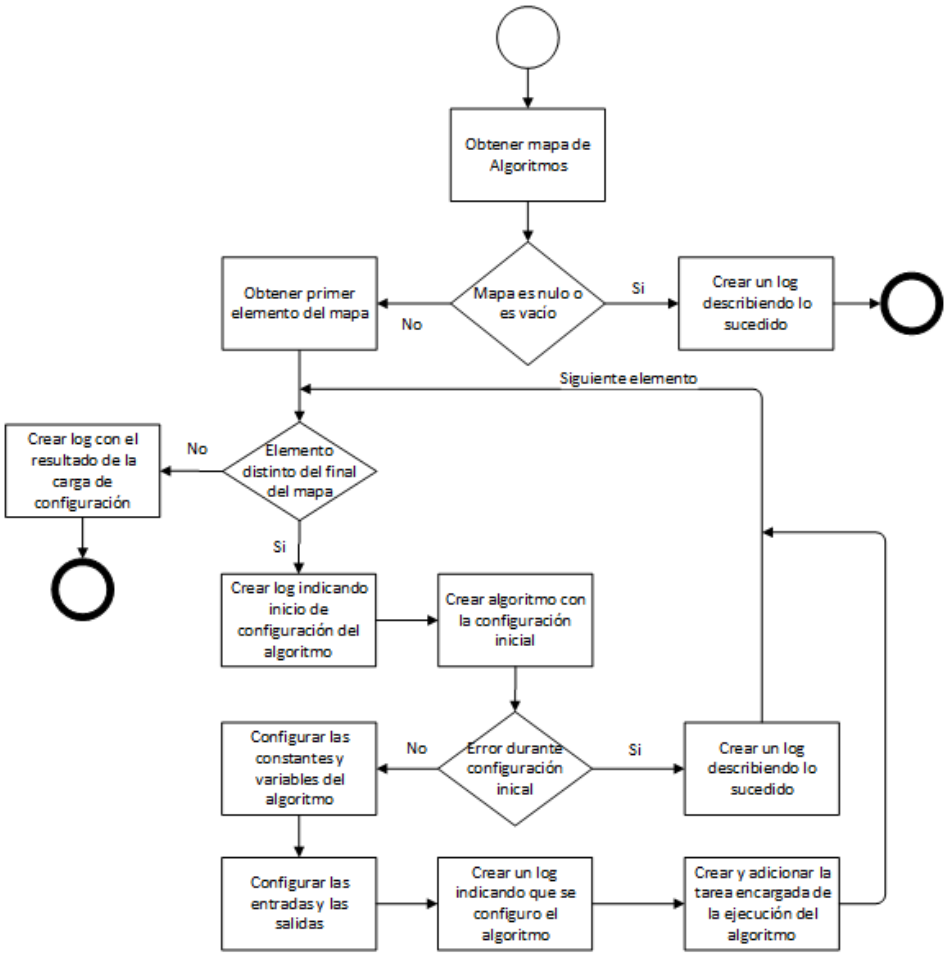


Figura 20 Diagrama de flujo de la configuración de algoritmos.

Ejecución del Algoritmo

Una vez realizada la configuración de los algoritmos se continúa con la ejecución de los que se configuraron correctamente; proceso de ejecución que se describe a continuación:

Pseudocódigo

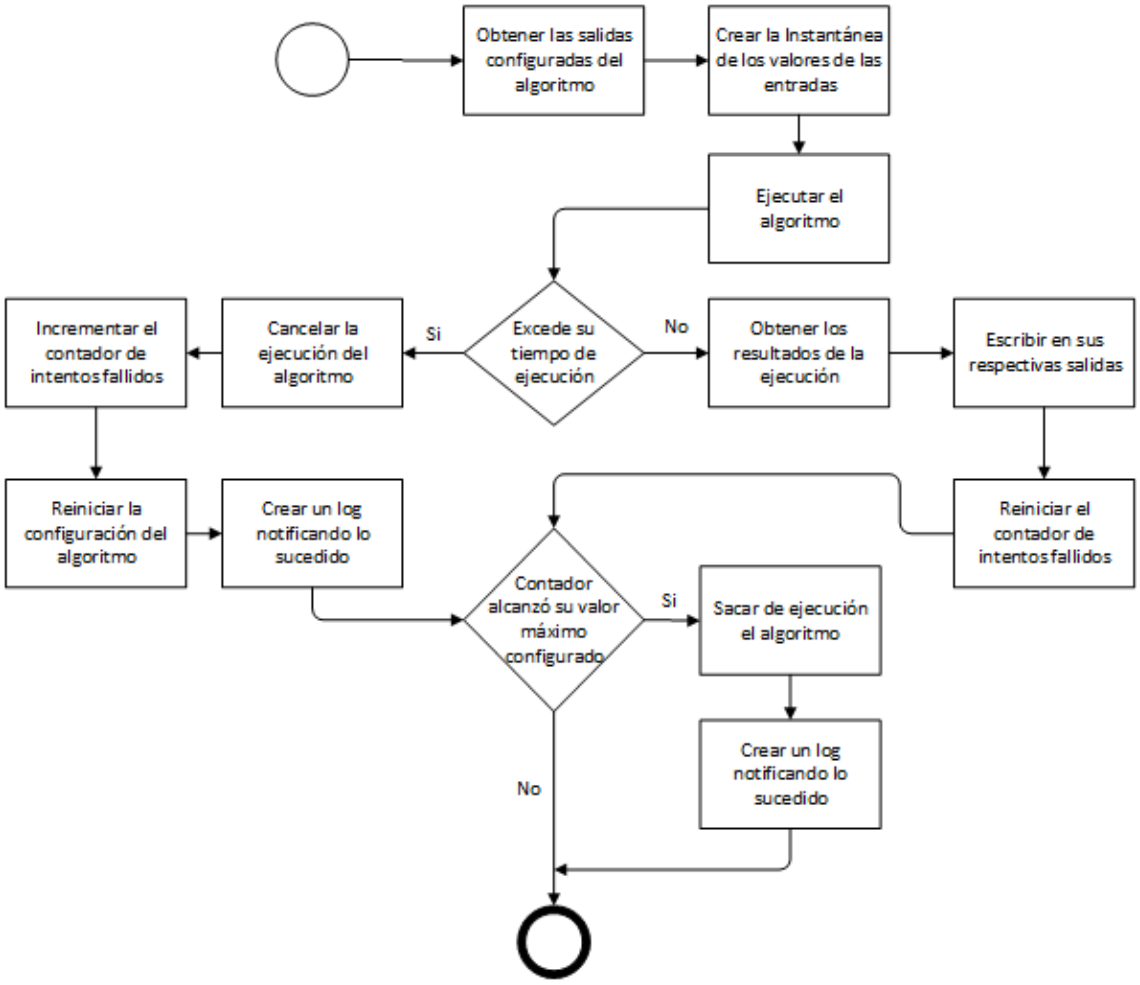


Figura 21 Diagrama de flujo de la ejecución del algoritmo.

3.2 Pruebas

Una vez que se ha generado el código comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos. (23)

A las funcionalidades de la presente solución se le realizaron pruebas unitarias de caja negra y pruebas de rendimiento.

Pruebas unitarias

Se realizaron con el objetivo de probar las unidades individuales del sistema, unidades que en la programación procedural pueden ser un programa individual, una función, un procedimiento, entre otros, mientras que en la programación orientada a objetos una unidad puede ser un método, una clase abstracta, una clase base e incluso una clase hija. Teniendo como propósito validar que cada unidad del software realiza su función como se ha diseñado.

Pruebas de caja negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se centran en los requisitos funcionales del software. Es decir, las técnicas de pruebas caja negra permiten definir conjuntos de condiciones de entrada que van a ejercitar plenamente a los requisitos funcionales del programa. (23)

Pruebas de rendimiento

Se diseñaron para asegurar que el subsistema pueda procesar su carga esperada. También para validar y verificar atributos de calidad tales como la escalabilidad y uso de los recursos. (18)

Herramientas utilizadas

En la realización de las pruebas al subsistema de algoritmos se utilizaron las herramientas siguientes:

- ModbusPal v1.6b y Modsim32 v4.A00-04: Se emplearon para la simulación de dispositivos de campos. Ambos crean dispositivos virtuales que se comunican por el protocolo Modbus en su variante TCP, y se les pueden asociar funciones a las variables para la simulación de valores o cambiar los valores de forma manual.

- Algoritmos de pruebas: Se desarrollaron para simular el proceso de carga y ejecución con diferentes características que pueden ser empleados en las industrias. Entre ellos se encuentran la implementación de uno con un ciclo infinito para el caso de prueba de tiempo excedido, otros que varían entre la cantidad de salidas y las de entradas, y para el desarrollo de un algoritmo PID se empleó como

base la implementación expuesta en Embedded Control Systems in C/C++ (ver anexo 1) (24).

Ambiente de pruebas

Para realizar las pruebas se emplearon dos ordenadores personales. El primer ordenador (PC1) para la puesta en ejecución del ModbusPal y el Modsim32, mientras que en el segundo (PC2) para la puesta en marcha del módulo de adquisición con la solución integrada al mismo. Las pruebas fueron realizadas sobre el sistema operativo GNU/Linux Debian Squeeze en su versión 6.0.9.

Tabla 3 Recursos de hardware para las pruebas.

Ordenadores	Hardware
PC1	Intel(R) Core(TM) 2 Duo E4500 a 2.20GHz y 1GB de memoria principal (RAM)
PC2	Intel(R) Core(TM) i3-2100 CPU a 3.10GHz y 2GB de memoria principal (RAM)

Los algoritmos fueron configurados con entradas y las salidas se relacionaron a variables del sistema SCADA. En el caso particular del algoritmo PID se le configuraron además otros parámetros de entradas, como el valor máximo (10000) y mínimo (0) de su salida, el error (0.03), el valor que se quiere alcanzar (1000) con su ejecución, entre otros.

3.2.1 Descripción de los casos de pruebas

A continuación se documentan los casos de pruebas diseñados para validar el correcto funcionamiento de la solución según los requisitos funcionales definidos.

Tabla 4 Prueba a la carga de configuración.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Carga de configuración.	EC 1.1: Cargar configuración.	Esta funcionalidad realiza la cargar de la configuración del subsistema.	Se lee la configuración del módulo configuración. Se configuran los algoritmos llegados de configuración. Los algoritmos se guardan para posterior ejecución.

ID del escenario	Escenario	ID de Algoritmos	Resultado esperado del sistema	Resultado de la prueba
EC 1.1	Cargar configuración.	81 489 193 676 111	Se espera que una vez leída la configuración cargue los cinco algoritmos y al final cree un log con la cantidad de algoritmos cargados correctamente y los que no se pudieron cargar.	El sistema cargó los cinco algoritmos leídos de configuración y al finalizar creó el log correctamente de lo sucedido durante el proceso de configuración (ver anexo 2).
	Cargar configuración con valores	193 666	Se espera que una vez leída la configuración cargue	El sistema cargó los dos algoritmos

	erróneos.	111	los tres algoritmos que le llegan configurados, de los cuales uno no podrá ser cargado (el 666) pues la dirección del algoritmo es incorrecta, lo cual será reportado en un log, al final debe crear un log con la cantidad de algoritmos cargados correctamente y los que no se pudieron cargar.	configurados correctamente y creo el log correspondiente al algoritmo que no pudo configurar, al finalizar creó el log correctamente de lo sucedido durante el proceso de configuración.
	Cargar configuración en blanco		El resultado esperado es la creación de un log que indica cuantos algoritmos fueron cargados, en este caso en particular se debe dejar plasmado de que no se recibió ningún algoritmo a cargar.	El sistema creó el log informando de que no existían algoritmos a cargar (ver anexo 3).

Tabla 5 Pruebas a la ejecución de los algoritmos.

Nombre de la	**Escenarios de**	**Descripción de la**	**Flujo central**

sección	**la sección**	**funcionalidad**	
Ejecución de Algoritmos.	EC 2.1: Ejecutar algoritmos de una sola entrada y una sola salida.	Esta funcionalidad permite la ejecución de algoritmos de una sola entrada y una sola salida.	Se realiza la actualización de la entrada relacionada al punto del SCADA, creando una instantánea del valor de entrada. Se ejecuta el algoritmo y controla que no exceda el tiempo de ejecución estimado previamente configurado. Una vez terminada la ejecución se lee la salida del algoritmo. Se actualiza el valor asociado a la salida en el sistema.
	EC 2.2: Ejecutar algoritmos de múltiples entradas y una sola salida.	Esta funcionalidad permite la ejecución de algoritmos de múltiples entradas y una sola salida.	Se realiza la actualización de las entradas relacionadas con los puntos del SCADA, creando una instantánea de los valores de las entradas. Se ejecuta el algoritmo

			y controla que no exceda el tiempo de ejecución estimado previamente configurado. Una vez terminada la ejecución se lee la salida del algoritmo. Se actualiza el valor asociado a la salida en el sistema.
	EC 2.3: Ejecutar algoritmos de una sola entrada y múltiples salidas.	Esta funcionalidad permite la ejecución de algoritmos de una sola entrada y múltiples salidas.	Se realiza la actualización de la entrada relacionada al punto del SCADA, creando una instantánea del valor de entrada. Se ejecuta el algoritmo y controla que no exceda el tiempo de ejecución estimado previamente configurado. Una vez terminada la ejecución se leen las salidas del algoritmo. Se actualizan las respectivas salidas

			asociadas en el sistema.
	EC 2.4: Ejecutar algoritmos de múltiples entradas y múltiples salidas.	Esta funcionalidad permite la ejecución de algoritmos de múltiples entradas y múltiples salidas.	Se realiza la actualización de las entradas relacionadas con los puntos del SCADA, creando una instantánea de los valores de las entradas. Se ejecuta el algoritmo y controla que no exceda el tiempo de ejecución estimado previamente configurado. Una vez terminada la ejecución se leen las salidas del algoritmo. Se actualizan las respectivas salidas asociadas en el sistema.

ID del escenario	Escenario	ID de Algoritmos	Resultado esperado del sistema	Resultado de la prueba
EC 2.1	Ejecutar algoritmos de una sola	81	El sistema debe ser capaz de ejecutar	El sistema antes de comenzar con la ejecución creó la

	entrada y de una sola salida.		algoritmos de una sola entrada y de una sola salida. Durante el proceso debe crear una instantánea de la entrada del algoritmo, controlar que no exceda el tiempo de ejecución estimado y una vez terminada la ejecución tiene que actualizar la salida asociada.	instantánea del valor de la entrada, luego prosiguió con la ejecución del algoritmo, el cual se ejecutó en tiempo, una vez terminada la ejecución leyó la salida del mismo y continuó con la actualización de la salida correspondiente.
EC 2.2	Ejecutar algoritmos de múltiples entradas y de una sola salida.	193	El sistema debe ser capaz de ejecutar algoritmos de múltiples entradas, y de una sola salida. Durante el proceso debe crear una instantánea de las entradas del	El sistema antes de comenzar con la ejecución creó la instantánea de los valores de las entradas, luego prosiguió con la ejecución del algoritmo, el cual se ejecutó en tiempo, una vez terminada la ejecución leyó la salida del mismo y

			algoritmo, controlar que no exceda el tiempo de ejecución estimado y una vez terminada la ejecución del mismo tiene que actualizar la salida asociada.	continuó con la actualización de la salida correspondiente.
EC 2.3	Ejecutar algoritmos de una sola entrada y múltiples salidas.	111	El sistema debe ser capaz de ejecutar algoritmos de tipo de una sola entrada y múltiples salidas. Durante el proceso debe crear una instantánea de la entrada del algoritmo, controlar que no exceda el tiempo de ejecución estimado y una vez terminada la ejecución del	El sistema antes de comenzar con la ejecución creó la instantánea del valor de la entrada, luego prosiguió con la ejecución del algoritmo, el cual se ejecutó en tiempo, una vez terminada la ejecución leyó las salidas del mismo y continuó con la actualización de las salidas correspondientes.

			mismo tiene que actualizar las salidas asociadas.	
EC 2.4	Ejecutar algoritmos de múltiples entradas y múltiples salidas.	676 193	El sistema debe ser capaz de ejecutar algoritmos de múltiples entradas y múltiples salidas. Durante el proceso debe crear una instantánea de las entradas del algoritmo, controlar que no exceda el tiempo de ejecución estimado y una vez terminada la ejecución del mismo tiene que actualizar las salidas asociadas.	El sistema antes de comenzar con la ejecución creó la instantánea de los valores de las entradas, luego prosiguió con la ejecución del algoritmo, el cual se ejecutó en tiempo, una vez terminada la ejecución leyó las salidas del mismo y continuó con la actualización de las salidas correspondientes.

Tabla 6 Pruebas al control del tiempo de ejecución.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Control del tiempo de ejecución de los algoritmos	EC 3.1 Ejecución de algoritmos que excedan el tiempo estimado.	Esta funcionalidad vela porque los algoritmos no sobrepasen el tiempo estimado de ejecución configurado.	Una vez ejecutado el algoritmo. Se controla que no exceda el tiempo estimado de ejecución. Si se agotada el tiempo estimado para la ejecución del algoritmo, se cancela, se reinicia a la configuración inicial y se incrementa el contador de intentos fallidos. En caso contrario se reinicia a cero el contador. Si el contador alcanza su valor máximo configurado se elimina de ejecución la tarea.

ID del escenario	Escenario	ID de Algoritmos	Resultado esperado del sistema	Resultado de la prueba
EC 3.1	Ejecución de	753	El sistema debe	El sistema ante el

	algoritmos que excedan el tiempo estimado.	85 486 15	ser capaz de controlar que los algoritmos no excedan el tiempo configurado para su ejecución. En caso afirmativo, cancelarlo, reiniciarlo, cargarle la configuración inicial, e incrementar el contador de intentos fallidos. Si el contador llegara a su máximo valor establecido, se debe eliminar la tarea de ejecución y generar un log relacionado con lo sucedido.	tiempo excedido de ejecución los canceló (ver anexo 6), reinició e incrementó el contador de intentos. Una vez alcanzado el valor máximo de intentos se eliminó la tarea de ejecución y se creó el log correspondiente (ver anexo 5).

3.2.2 Resultado de las pruebas realizadas a las funcionalidades

El proceso de pruebas contó con cuatro iteraciones. Detectándose un total de 20 no conformidades de las cuales 5 no procedieron.

Dichas no conformidades fueron detectadas en la siguiente distribución:

En una primera iteración se detectaron dos críticas, cuatro medias y tres bajas, para un total de nueve no conformidades.

Para la segunda iteración se detectaron una crítica, tres medias y cuatro bajas, para un total de ocho no conformidades.

En la tercera iteración fueron detectadas una media y dos bajas, para un total de tres no conformidades.

En la última iteración no fueron detectadas no conformidades.

La clasificación de severidad que se manejó en la realización de las pruebas fue la siguiente:

- **Critica:** Se considera como una no conformidad que compromete todo el subsistema de algoritmos, causando una fuga de memoria o la interrupción del software.

- **Media:** Se tomó como una no conformidad que afecta el subsistema de algoritmos sin comprometer la ejecución del software, dígase la ejecución de manera atípica de algoritmos o funcionalidades deseadas.

- **Baja:** Se definió como una no conformidad que no están relacionadas directamente a las funcionalidades descritas, sino más bien que compromete con la calidad del producto, dígase errores ortográficos y/o omisión de descripciones de errores.

A partir de la gráfica presentada y de la clasificación de las no conformidades se puede apreciar que las no conformidades críticas representaron solo el 15% del total. Las de severidad media representaron un 40%. Mientras que las de severidad baja representaron un 45%.

3.2.3 Pruebas de rendimiento

Se diseñaron dos escenarios, el primero compuesto por seis casos de pruebas con el objetivo de medir los tiempos (microsegundos) en que se cargaban los algoritmos; mientras que en el segundo se definieron cuatro casos teniendo como meta determinar la capacidad máxima recomendada de algoritmos en ejecución.

Para el cálculo del tiempo de respuesta obtenido (To) del primer escenario se empleó la fórmula siguiente:

$$To = \sum_{i=0}^{n} (Tf_i - Ti_i)/\mathrm{n}$$

Donde Tf es la marca de tiempo final, Ti la inicial y n el número de veces que se realizó la prueba.

En el primer escenario, se toma Ti en el momento que inicia el proceso de configuración y Tf cuando termina.

Mientras que para la realización de la prueba en el segundo escenario se utilizó la siguiente:

$$To = \sum_{i=0}^{n} (T_i - T_{i-1})/\mathrm{n}$$

Usándose T como instante en que se ejecuta el algoritmo, y n la cantidad de veces que se ejecutó el mismo.

3.2.4 Resultado de las pruebas de rendimiento

A continuación se muestran en gráficas los resultados obtenidos para una mayor comprensión:

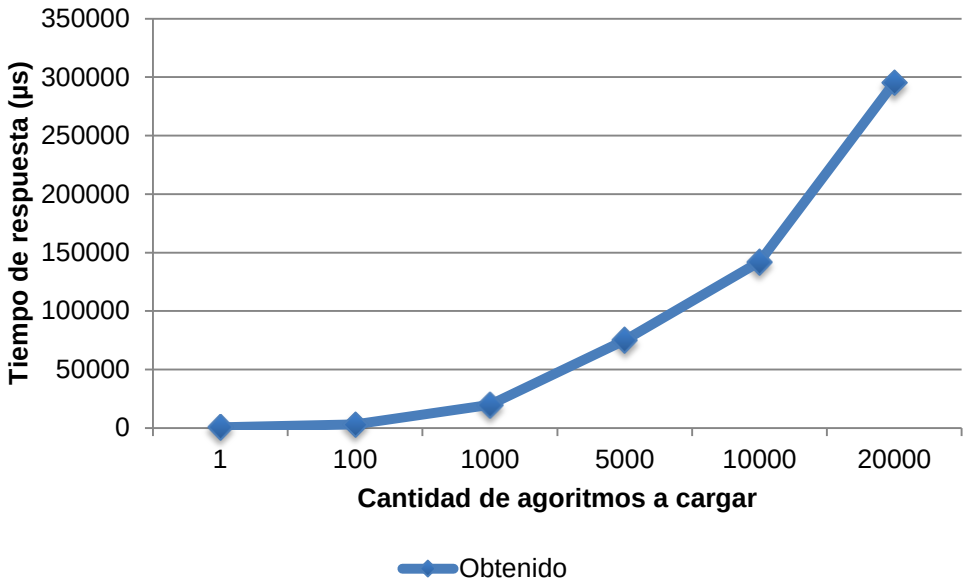


Figura 22 Gráfica del tiempo de carga de algoritmos.

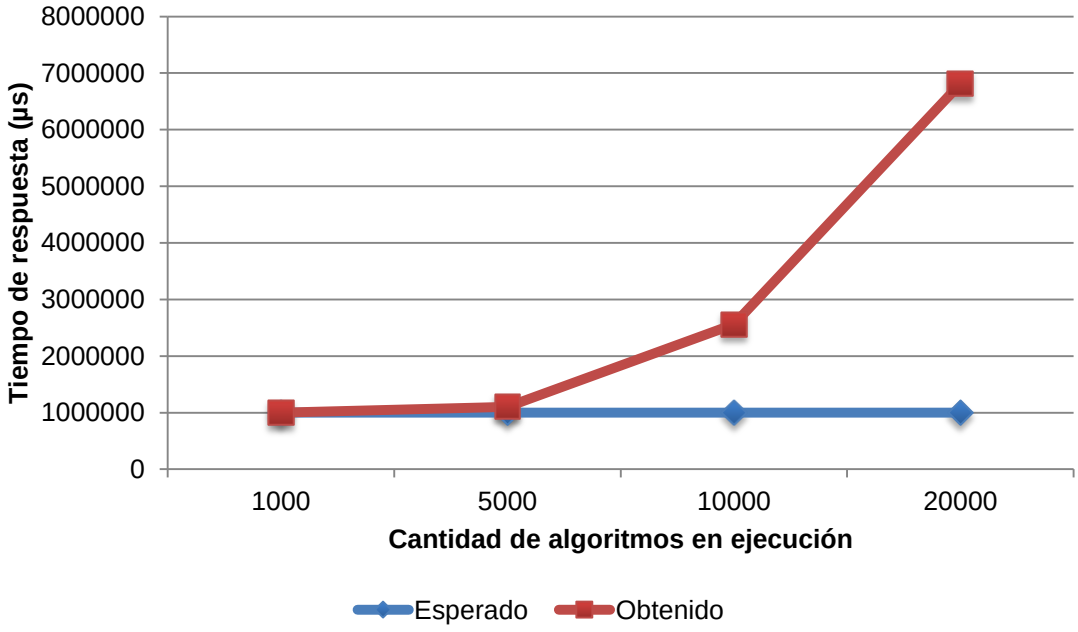


Figura 23 Gráfica de afectación del periodo de ejecución.

En la Figura 17 se aprecian los tiempos de respuestas obtenidos contra la cantidad de algoritmos a cargar para su ejecución. A su vez en la Figura 18 se muestran los tiempos de respuesta esperados y obtenidos contra la cantidad de algoritmos en ejecución.

Las pruebas arrojan una relación proporcional entre la cantidad de algoritmos y el tiempo que demoran en cargarse y ejecutarse. O sea a mayor cantidad de algoritmos, mayor tiempo de espera para su próxima ejecución.

Conclusiones

Al culminar la presente investigación tras la etapa de pruebas y el análisis de los resultados, se arribaron a las siguientes conclusiones sobre dicho desarrollo:

- Posibilita la incorporación al SCADA Guardián del ALBA, de algoritmos que se ejecutarán cumpliendo con las restricciones de tiempo presentes en los escenarios de despliegue del sistema.

- Brinda la flexibilidad de permitir la incorporación y ejecución, de múltiples tipos de algoritmos.

- Contribuye a que el SCADA Guardián del ALBA sea más aplicable, competitivo y a bajar los costos de implantación en aplicaciones de control automático no crítico.

Recomendaciones

Con el fin de mejorar las funcionalidades desarrolladas en el presente trabajo se recomienda:

- Dar soporte a la configuración en caliente del subsistema.
- Integrar el subsistema a las nuevas versiones del SCADA GALBA.

Referencias Bibliográficas

1. **Farlex.** The Free Dictionary. [Online] En Línea. http://www.thefreedictionary.com/control.

2. **Moreno, Mauricio Améstegui.** *Apuntes de control PID.* La Paz : s.n., 2011.

3. **Cevallos, María I. Hernández and Marcalla, Denis A. Ledesma.** *Desarrollo de un sistema SCADA para la medición de voltajes con sistemas embebidos para el laboratorio de mecatrónica de la Facultad de Mecánica.* 2010.

4. **Ogata, Katsuhiko.** *Ingeniería de control moderna 3ra edición.* s.l. : Prentice Hall HISPANOAMERICANA, S.A., 1995.

5. **Smith, Carlos A. and Corripio, Armando B.** *Control Automático de procesos. Teoría y Práctica.* s.l. : Editorial Limusa, 1991. 968-18-3791-6.

6. Tecnología e Informática Grado10. [Online] En Línea. https://sites.google.com/site/presentacioncartagoti10/definicion-de-algoritmos.

7. *Design and analysis of differential evolution algorithm based automatic generation control for interconnected power system.* **Rout, Umesh Kumar, Sahu, Rabindra Kumar and Panda, Sidhartha.** Odisha : Ain Shams Engineering Journal, 2012.

8. **Vázquez, Moisés Herrera, Hernández, Yaneisy and Pérez, Jaime Fardales.** *Recolección y Manejadores en el Guardián del ALBA.* 2008.

9. **Penin, Aquilino Rodríguez.** *Sistemas SCADA 2da edición.* Barcelona : MARCOMBO, 2007.

10. **Vázquez, Moisés Herrera, Hernández, Yaneisy and Pérez, Jaime Fardales.** *Introducción a la Arquitectura del Guardián del ALBA.* 2008.

11. **Pérez, Daimeris Velasco and Mir, Yuniesky Orlando Vasconcelo.** *Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA.* 2010.

12. **Fariñas, Hebert Hernández.** *Versión Miranda R2 del módulo de adquisición del SCADA "Guardián del ALBA".* 2013.

13. **AG, Siemens.** *SIMATIC WinCC System Description.* 2008.

14. **Progea.** *MOVICON Ver.9.1 User Manual.* 2002.

15. **Corporation, Wonderware.** *Wonderware® FactorySuiteTM InTouch®.* 1999.

16. *OpenSCADA v. 0.8.0.* 2012.

17. **Tamayo, Juan Carlos González.** *Desarrollo de funcionalidades al módulo de procesamiento del SCADA UX.* 2011.

18. **Sommerville, Ian.** *Software Engineering 8th ed.* 2007.

19. **Rumbaugh, James, Jacobson, Ivar and Booch, Grady.** *El Lenguage Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley.

20. cplusplus. [Online] En línea. http://www.cplusplus.com/info/description/.

21. **Varela, Angel Omelio George.** *Aislamiento de procesos en la obtención de puntos calculados del SCADA UX.* Habana : s.n., 2012.

22. **Lorenzo, Ariel Chávez.** *ESCMGA-SGA-0120_1 Estándares de codificación.* 2010.

23. **Pressman, Roger S.** *Software Engineer: A Practitioner's Approach.* s.l. : The McGraw-Hill, 2010. 978-0-07-337-597.

24. **Ledin, Jim.** *Embedded Control Systems in C/C++: An Introduction for Software Developers Using MATLAB.* s.l. : CMP Books, 2004. 1578201276.

Bibliografía

1. **Farlex.** The Free Dictionary. [En línea] En Línea. http://www.thefreedictionary.com/control.

2. **Moreno, Mauricio Améstegui.** *Apuntes de control PID.* La Paz : s.n., 2011.

3. **Cevallos, María I. Hernández y Marcalla, Denis A. Ledesma.** *Desarrollo de un sistema SCADA para la medición de voltajes con sistemas embebidos para el laboratorio de mecatrónica de la Facultad de Mecánica.* 2010.

4. **Ogata, Katsuhiko.** *Ingeniería de control moderna 3ra edición.* s.l. : Prentice Hall HISPANOAMERICANA, S.A., 1995.

5. **Smith, Carlos A. y Corripio, Armando B.** *Control Automático de procesos. Teoría y Práctica.* s.l. : Editorial Limusa, 1991. 968-18-3791-6.

6. Tecnología e Informática Grado10. [En línea] En Línea. https://sites.google.com/site/presentacioncartagoti10/definicion-de-algoritmos.

7. *Design and analysis of differential evolution algorithm based automatic generation control for interconnected power system.* **Rout, Umesh Kumar, Sahu, Rabindra Kumar y Panda, Sidhartha.** Odisha : Ain Shams Engineering Journal, 2012.

8. **Vázquez, Moisés Herrera, Hernández, Yaneisy y Pérez, Jaime Fardales.** *Recolección y Manejadores en el Guardián del ALBA.* 2008.

9. **Penin, Aquilino Rodríguez.** *Sistemas SCADA 2da edición.* Barcelona : MARCOMBO, 2007.

10. **Vázquez, Moisés Herrera, Hernández, Yaneisy y Pérez, Jaime Fardales.** *Introducción a la Arquitectura del Guardián del ALBA.* 2008.

11. **Pérez, Daimeris Velasco y Mir, Yuniesky Orlando Vasconcelo.** *Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA.* 2010.

12. **Fariñas, Hebert Hernández.** *Versión Miranda R2 del módulo de adquisición del SCADA "Guardián del ALBA".* 2013.

13. **AG, Siemens.** *SIMATIC WinCC System Description.* 2008.

14. **Progea.** *MOVICON Ver.9.1 User Manual.* 2002.

15. **Corporation, Wonderware.** *Wonderware® FactorySuiteTM InTouch®.* 1999.

16. *OpenSCADA v. 0.8.0.* 2012.

17. **Tamayo, Juan Carlos González.** *Desarrollo de funcionalidades al módulo de procesamiento del SCADA UX.* 2011.

18. **Sommerville, Ian.** *Software Engineering 8th ed.* 2007.

19. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El Lenguage Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley.

20. cplusplus. [En línea] En línea. http://www.cplusplus.com/info/description/.

21. **Varela, Angel Omelio George.** *Aislamiento de procesos en la obtención de puntos calculados del SCADA UX.* Habana : s.n., 2012.

22. **Lorenzo, Ariel Chávez.** *ESCMGA-SGA-0120_1 Estándares de codificación.* 2010.

23. **Pressman, Roger S.** *Software Engineer: A Practitioner's Approach.* s.l. : The McGraw-Hill, 2010. 978-0-07-337-597.

24. **Ledin, Jim.** *Embedded Control Systems in C/C++: An Introduction for Software Developers Using MATLAB.* s.l. : CMP Books, 2004. 1578201276.

25. **Ogata, Katsuhiko.** *Sistema de Control en Tiempo Discreto 2da edición.* s.l. : Prentice Hall HISPANOAMERICANA, S.A., 1996.

26. *Predicción para control: Una panorámica del control de procesos con retardo.* **Normey-Rico, Julio E. y Camacho, Eduardo F.** 4, s.l. : Revista Iberoamericana de Automática e Informática Industrial, 2006, Vol. 3.

27. **Lara, Carlos Barcala.** *Aplicación de un Control Predictivo-Adaptativo al AGC (Control Automático de Generación) en un sistema eléctrico de potencia.*

28. **Vázquez, Moisés Herrera, Hernández, Yaneisy y Pérez, Jaime Fardales.** *Base de Datos de Tiempo Real en el "GUARDIÁN DEL ALBA".* 2008.

29. **Peralta González, Yunior y Tellería Martínez, Ana Silvia.** *Módulo de adquisición para un sistema SCADA.* 2011.

30. **Dorf, Richard C. y Bishop, Robert H.** *Modern Control Systems 12Th Ed.* s.l. : Prentice Hall, 2011.

31. **Invensys Systems, Inc.** *InTouch ® HMI and ArchestrA ® Integration Guide.* 2007.

32. **León, Rolando Alfredo Hernández y González, Sayda Coello.** *El proceso de investigación científica.* s.l. : La Editorial Universitaria, 2011. 978-959-16-1307-3.

33. **Gamma, Erich, y otros, y otros.** *Design Patterns: Elements of Reusable Object Oriented Software.* s.l. : Addison Wesley, 1995.

34. **SPARX SYSTEMS.** Plataforma avanzada de modelado y diseño. [En línea] En línea. http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.

35. **Software Testing Fundamentals.** Software Testing Fundamentals. [En línea] En Línea. http://softwaretestingfundamentals.com.

Anexos

```
real PID_Controller::Update(real error)
{
    // Set q to 1 if the error magnitude is below
    // the threshold and 0 otherwise
    real q;
    if (fabs(error) < m_error_thresh)
        q = 1;
    else
        q = 0;

    // Update the error integral
    m_integral += m_h*q*error;

    // Compute the error derivative
    real deriv;
    if (!m_started)
    {
        m_started = true;
        deriv = 0;
    }
    else
        deriv = (error - m_prev_error) * m_inv_h;

    m_prev_error = error;

    // Return the PID controller actuator command
    return m_kp*(error + m_ki*m_integral + m_kd*deriv);
}
```

Anexo 1 Implementación base del algoritmo PID.

```
INFO  2014-05-20 11:01:14,469 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(87)
      [ ALGORITHM ]: Configurando el algoritmo # 1 con nombre MyTestPID y direccion /media/truecrypt2/Algorithm_PID/De
bug/libAlgorithm_PID.so
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(112)
      [ ALGORITHM ]: Configurando constantes y variables del algoritmo # 1
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: K Valor: 0.1
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: Ti Valor: 0.4
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: Td Valor: 0.2
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: Tt Valor: 10
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: N Valor: 10
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: b Valor: 1
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: ulow Valor: 0
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: uhigh Valor: 10000
INFO  2014-05-20 11:01:14,505 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: h Valor: 0.03
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(163)
      [ ALGORITHM ]: Atributo: uc Valor: 1000
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(116)
      [ ALGORITHM ]: Configurando entradas del algoritmo # 1
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(182)
      [ ALGORITHM ]: Entrada: input1 Id: 36554434
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(182)
      [ ALGORITHM ]: Entrada: input2 Id: 36554434
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(120)
      [ ALGORITHM ]: Configurando salidas del algoritmo # 1
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(201)
      [ ALGORITHM ]: Salida: output1 Id: 36554435
INFO  2014-05-20 11:01:14,506 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(125)
      [ ALGORITHM ]: El Algoritmo con Id 1 esta configurado y listo para su ejecucion.
```

Anexo 2 Detalles del algoritmo cargado.

```
INFO  2014-05-20 12:55:07,127 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(150)
      [ ALGORITHM ]: No se recibieron algoritmos desde configuracion.
```

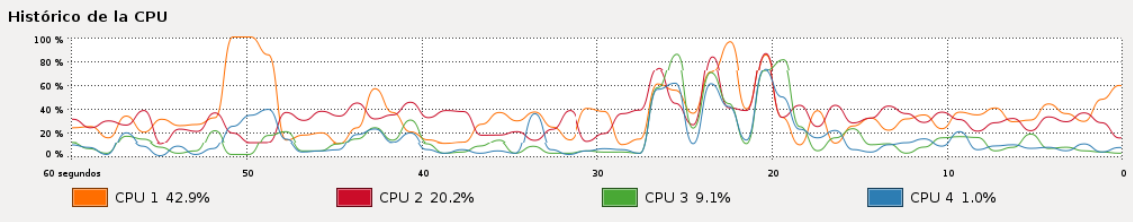
Anexo 3 No se cargaron algoritmos desde configuración.

```
      [ ALGORITHM ]: Salida: output1 Id: 36554445034
INFO  2014-05-20 11:01:16,042 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(125)
      [ ALGORITHM ]: El Algoritmo con Id 5000 esta configurado y listo para su ejecucion.
INFO  2014-05-20 11:01:16,042 [acquisitionAlgorithmsLog] configureFacade/ConfigureFacade.cpp(145)
      [ ALGORITHM ]: Se configuraron correctamente 5000 algoritmos y no fueron configurados 0
```

Anexo 4 Algoritmo listo para su ejecución.

```
INFO  2014-05-20 12:46:52,177 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(87)
      [ ALGORITHM ]: Se reinicia el algoritmo 100
FATAL 2014-05-20 12:46:53,950 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(76)
      [ ALGORITHM ]: Tiempo de espera agotado en la ejecucion del algoritmo 20
INFO  2014-05-20 12:46:53,950 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(87)
      [ ALGORITHM ]: Se reinicia el algoritmo 20
ERROR 2014-05-20 12:46:53,955 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(139)
      [ ALGORITHM ]: Reintentos agotados para el algoritmo # 20. Se finaliza la ejecucion.
FATAL 2014-05-20 12:46:54,051 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(76)
      [ ALGORITHM ]: Tiempo de espera agotado en la ejecucion del algoritmo 40
INFO  2014-05-20 12:46:54,051 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(87)
      [ ALGORITHM ]: Se reinicia el algoritmo 40
ERROR 2014-05-20 12:46:54,051 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(139)
      [ ALGORITHM ]: Reintentos agotados para el algoritmo # 40. Se finaliza la ejecucion.
FATAL 2014-05-20 12:46:54,153 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(76)
      [ ALGORITHM ]: Tiempo de espera agotado en la ejecucion del algoritmo 60
INFO  2014-05-20 12:46:54,153 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(87)
      [ ALGORITHM ]: Se reinicia el algoritmo 60
ERROR 2014-05-20 12:46:54,153 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(139)
      [ ALGORITHM ]: Reintentos agotados para el algoritmo # 60. Se finaliza la ejecucion.
FATAL 2014-05-20 12:46:54,270 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(76)
      [ ALGORITHM ]: Tiempo de espera agotado en la ejecucion del algoritmo 80
INFO  2014-05-20 12:46:54,270 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(87)
      [ ALGORITHM ]: Se reinicia el algoritmo 80
ERROR 2014-05-20 12:46:54,270 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(139)
      [ ALGORITHM ]: Reintentos agotados para el algoritmo # 80. Se finaliza la ejecucion.
FATAL 2014-05-20 12:46:55,177 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(76)
      [ ALGORITHM ]: Tiempo de espera agotado en la ejecucion del algoritmo 100
INFO  2014-05-20 12:46:55,177 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(87)
      [ ALGORITHM ]: Se reinicia el algoritmo 100
ERROR 2014-05-20 12:46:55,177 [acquisitionAlgorithmsLog] tasks/ExecuteAlgorithmTask.cpp(139)
      [ ALGORITHM ]: Reintentos agotados para el algoritmo # 100. Se finaliza la ejecucion.
```

Anexo 5 Reintentos agotados.



Anexo 6 Ejecución de algoritmos.