REPÚBLICA DE CUBA
UNIVERSIDAD DE ORIENTE
FACULTAD DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE CONTROL AUTOMÁTICO

TESIS DE MAESTRÍA

DESARROLLO DE UN SISTEMA SCADA FLEXIBLE PARA UN LABORATORIO DE MEDICIONES

Tesis en opción al Título de Máster en Ciencias

Autor: Ing. Alfredo Pino Escalona

Tutor: Dr. Israel Benítez Pina

Consultantes: Prof. Dr. Jose Reinaldo Silva (EP-USP, Sao Paulo, Brasil)
Prof. Dr. Antoni Sudria Andreu (CITCEA-UPC, Barcelona, España)

Enero de 2015

A mi pequeño Emilito…

A todas esas personas que me han apoyado en esta odisea: Familia, amigos y compañeros del GERA, muchas gracias.

A mi tutor, por soportar mis desapariciones, eternamente agradecido.

RESUMEN

En este trabajo se desarrolla un sistema SCADA para el Laboratorio de Mediciones del Grupo de Energías Renovables Aplicadas de la Universidad de Oriente. Caracterizado por opciones únicas de flexibilidad que lo diferencian de los conceptos clásicos, ya que facilitan la adaptación del sistema a las múltiples y cambiantes instalaciones a supervisar sin la necesidad de rediseñar su estructura funcional del mismo.

Para la implementación del proyecto se desarrollaron las ideas conceptuales tomando como base el esquema de funcionamiento del centro en cuestión, lo que permitió obtener y validar los correspondientes modelos en Redes de Petri antes de efectuar la implementación en el ambiente de desarrollo LabWindows/CVI.

Finalmente la solución dada está compuesta por tres aplicaciones de software que envían o reciben información a una red de dispositivos remotos de E/S y facilitan al usuario crear sus propias pantallas personalizadas y almacenar la información deseada para su posterior procesamiento.

ABSTRACT

In this work a SCADA system for the Laboratory of Applied Renewable Energy Group at the *Universidad de Oriente* has been developed. Characterized by unique flexibility options, that differentiates it from the classical concepts, as they facilitate the adaptation of the system to the multiple and changing facilities to monitor without the need to redesign its functional structure.

To implement the project, conceptual ideas has been developed based on the scheme of operation of the center in question, which allowed to obtaining and validating the models in Petri nets prior to implementation in the development environment LabWindows / CVI.

Finally the given solution is composed of three software applications that send and receive information to remote I/O network devices, and facilitate to create your own custom displays and store the desired information for further processing.

INDICE GENERAL

INTRODUCCIÓN

En el actual ambiente de negocios altamente competitivos, la industria es desafiada por la demanda de una mayor productividad, calidad, seguridad y protección medioambiental. Los pequeños márgenes de ganancia y la modernización constante de los procesos de manufactura enfatizan la necesidad de integración y optimización de los sistemas de producción donde el papel de los centros de investigación y desarrollo de tecnologías es determinante en lograr estas metas.

A menudo los laboratorios de investigaciones se tornan ineficientes sin el empleo de modernas tecnologías para el control de procesos y el adecuado manejo de la información recopilada (Tommila, Juhani, & Lauri, 2005), donde los investigadores esperan obtener funcionalidades mejoradas a un costo razonable; como son el manejo del conocimiento e información en tiempo real, la integración de los sistemas de monitoreo con el funcionamiento de equipos, tener una alta disponibilidad de los sistemas y mejoras en la flexibilidad de los programas de software son ejemplos de requisitos cruciales que deben soportar los actuales sistemas tecnológicos para este tipo de centros.

De igual manera los especialistas que desarrollan las aplicaciones necesitan herramientas eficientes que ofrezcan mejores prestaciones y características flexibles, demandando una rápida respuesta de los fabricantes, que a la vez afrontan el reto de satisfacer las crecientes necesidades de los clientes, mientras adaptan las estructuras de los sistemas a los constantes cambios en el ambiente técnico y funcional.

Durante muchos años los sistemas de control integrados, inteligentes y flexibles han sido el objetivo de organizaciones de estandarización, consorcios industriales y grupos de investigación (Patel, 2004). Las soluciones, sin embargo, han sido difíciles de alcanzar, en parte debido a la complejidad del tema, y en parte por conflicto de intereses comerciales (Bailey & Wright, 2003).

Algunos puntos importantes que se trabajan en la actualidad respecto a este tema son por ejemplo, ¿cómo mejorar los sistemas para que se ajusten realmente a los requisitos del usuario final? (Pillai, Mehta, & Patel, 2012), ¿Cómo interconectar de manera fácil e instantánea productos de diferentes fabricantes? (OPC Foundation, 2014), y en mayor grado ¿cómo reusar las aplicaciones de software SCADA existentes? (Rodríguez, 2007), es decir,

obtener aplicaciones de software reconfigurables y flexibles que puedan ser aplicadas a diferentes entornos sin que esto redunde en cambios estructurales.

Este último tema implica un mayor desafío para los diseñadores, ya que desde las etapas donde se concibe la ingeniería del software se hace necesario no solo el empleo de diagramas y metodologías clásicas, sino que también se impone el uso de herramientas matemáticas formales como las Redes de Petri (PN), que permitan modelar y simular el comportamiento de los sistemas antes de proceder a la correspondiente implementación (Sanz, 2013), (Frey & Litz, 2000), (Parallel Systems Engineering, 2011). Aunque el uso de métodos formales a priori no garantiza un resultado ciento por ciento correcto, puede sin embargo, incrementar grandemente la comprensión del sistema revelando incongruencias, ambigüedades e imprecisiones que de otra manera podrían pasar desapercibidas (Clarke & Wing, 1996), (Mendling, 2009).

De esta manera y tomando como base lo antes planteado se define como **Problema de Investigación** la necesidad de disponer de sistemas flexibles de supervisión, control y adquisición de datos (SCADA) ajustados a los requerimientos de centros de investigación. Para dar solución al problema planteado, este trabajo se desarrollará tomando como **Objeto de investigación** los Sistemas SCADA.

En aras de llevar a vías prácticas la solución del problema planteado y tomando como base que en el Grupo de Energías Renovables Aplicadas (GERA) de la Universidad de Oriente existe un sistema implementado que necesita ser sometido a modificaciones con el objetivo de ajustarse a los parámetros antes mencionados, como son flexibilidad, disminución de ocurrencia de errores, eficiencia, etc. se traza el **Objetivo** de implementar un Sistema SCADA para el GERA ajustado a las características inherentes de flexibilidad y funcionalidad requeridas en centros de investigaciones. Enmarcando así este trabajo dentro del **Campo de Acción** del diseño de sistemas SCADA flexibles para centros de investigación, donde se plantea la **Hipótesis** de que si la implementación del sistema basada en modelos previamente diseñados arroja resultados donde se evidencie la disminución de ocurrencia de errores, así como mejoras en términos de eficiencia, rendimiento, facilidades de uso y flexibilidad, entonces quedarían sentadas las bases y principios para el desarrollo y diseño

de sistemas SCADAS flexibles para este tipo de centros, lográndose así solucionar el problema planteado.

Las tareas propuestas a llevar a cabo en esta investigación parten del hecho que en el GERA ya se dispone de una versión en funcionamiento, que fue desarrollada empleando el software CVI/Labwindows de la Corporación National Instruments. Pero como todo sistema, es susceptible a mejoras conceptuales y estructurales, además no fue modelado empleando ningún método formal antes de su implementación, esto ha dado al traste con algunas limitantes en sus facilidades de uso y rendimiento en la toma de mediciones que pueden ser solucionadas en la nueva versión del sistema que se propone en este trabajo.

Tareas de Investigación:

- Hacer un levantamiento general de las características del actual sistema SCADA instalado en el GERA así como de las características que debe cumplir el nuevo sistema.
- Hacer un análisis sobre el concepto de sistemas SCADAS flexibles y delimitar las principales características de estos.
- Formalizar una propuesta de SCADA flexible mediante el modelado de los diferentes bloques que componen el sistema, realizando su verificación y validación formal.
- Implementar el sistema prototipo, tomando como base los modelos desarrollados y validados previamente.
- Evaluar el desempeño del sistema aplicado a diferentes aplicaciones prácticas.

Técnicas y Métodos:

- Análisis de fuentes documentales.
- Técnicas y métodos empíricos: Observación y Entrevistas.
- Método de análisis y síntesis.
- Métodos experimentales: Diseño y Simulación.

Significación Práctica:

Este trabajo tiene una alta significación práctica aplicable a laboratorios de investigaciones donde se hace necesaria la adquisición de datos, debido a que se pretende obtener un sistema con características únicas de flexibilidad y versatilidad no desarrolladas hasta el

momento en la literatura publicada. Además el sistema que se propone constará con una solución de software definitiva que se podrá ajustar constantemente a las necesidades de toma de datos y acciones de control sin efectuar cambios en la programación del mismo.

Estructura y organización del Informe:

La presente investigación se encuentra organizada en dos capítulos. El Capítulo I está compuesto por cuatro epígrafes, donde se analizan las principales características del sistema SCADA instalado en el laboratorio del GERA, se hace un análisis de las particularidades requeridas para sistemas flexibles y reconfigurables, y finalmente se estudian los principales métodos y herramientas para solucionar problemas de modelado y programación de sistemas concurrentes.

En el Capítulo II, compuesto por cinco epígrafes se diseña la nueva propuesta de un sistema SCADA Flexible y se formaliza mediante el modelado, verificación, validación formal e implementación de cada parte que conforma el sistema. Finalmente se valida el correcto funcionamiento mediante la explotación y uso profesional.

CAPITULO I: SISTEMAS SCADA PARA LABORATORIOS

Introducción

Partiendo de la concepción del actual SCADA instalado en el GERA, se desarrolla en este capítulo un análisis de los requerimientos funcionales que debe cumplir el nuevo sistema, basándose principalmente en los conceptos referentes a SCADAS flexibles, enfocados desde el punto de vista del desarrollador y del usuario final. Estudiando además los métodos y herramientas para el desarrollo de este tipo de aplicaciones.

1.1 El sistema SCADA del Laboratorio del GERA

El Grupo de Energías Renovables Aplicadas de la Universidad de Oriente tiene como objeto principal el desarrollo, estudio, evaluación y análisis de las diferentes aplicaciones prácticas de las energías renovables. Basado en este, fue desarrollada la primera aplicación SCADA para el centro. Enfocándose principalmente a la toma de mediciones y ejecución de algunas acciones de control. Esta concepción inicial fue implementada a la misma medida que se desarrollaron los conceptos operativos del laboratorio de mediciones de este centro, por lo que el resultado hasta la actualidad es un sistema SCADA con múltiples modificaciones y adecuaciones que por ende genera aleatoriamente fallas y respuestas inestables.

Para la nueva concepción del sistema y en aras de una mejor adecuación a las características de un laboratorio de evaluación de tecnologías que funcionan con fuentes de energías renovables, se partirá de la descripción del procedimiento que se emplea en este centro para desarrollar estas tareas, seguido de un análisis de las tecnologías y técnicas que podrán ser empleadas.

1.1.1 Descripción del proceso de ensayos de laboratorio

El inicio del procedimiento de evaluación de un sistema en el laboratorio de este centro de investigación incluye la instalación o montaje de la tecnología en el polígono o áreas del centro; en caso de no ser viable esta posibilidad se valorará la opción de expandir el alcance del sistema de medición y control a lugares remotos mediante el empleo de enlaces telemáticos.

Una vez garantizado el alcance del sistema de medición y control hasta la instalación que se estudiará o evaluará. El investigador de conjunto con los especialistas en instrumentación y control del centro serán los encargados de instalar los sensores y actuadores que se asociarán al experimento. De manera autónoma el servidor deberá detectar los nuevos dispositivos instalados y el administrador generará una llave de software donde se asignarán los permisos y accesos a los dispositivos de E/S del sistema; esta le será entregada al responsable que desarrollará el experimento. Finalmente el investigador dispondrá de un software cliente que se enlazará con el servidor antes mencionado mediante una red de datos TCP-IP y tendrá acceso a tomar mediciones y/o a ejecutar acciones de control.

Todo el proceso de configuración, toma de datos y procesamiento de acciones de control correrá por parte del investigador o cliente del sistema, donde podrá visualizar datos y tendencias en tiempo real mediante diferentes indicadores en pantalla, construir mímicos personalizados, configurar lazos de control y almacenar los datos obtenidos en un único fichero en formato ASCII (mediciones, acciones de control, alarmas generadas, etc.) Un diagrama de flujo general está representado en la Figura 1.1.

Finalmente el investigador deberá procesar la información adquirida empleando disímiles métodos y técnicas, que por el grado de especificidad y variedad no se previó incluir dentro del alcance de este trabajo.

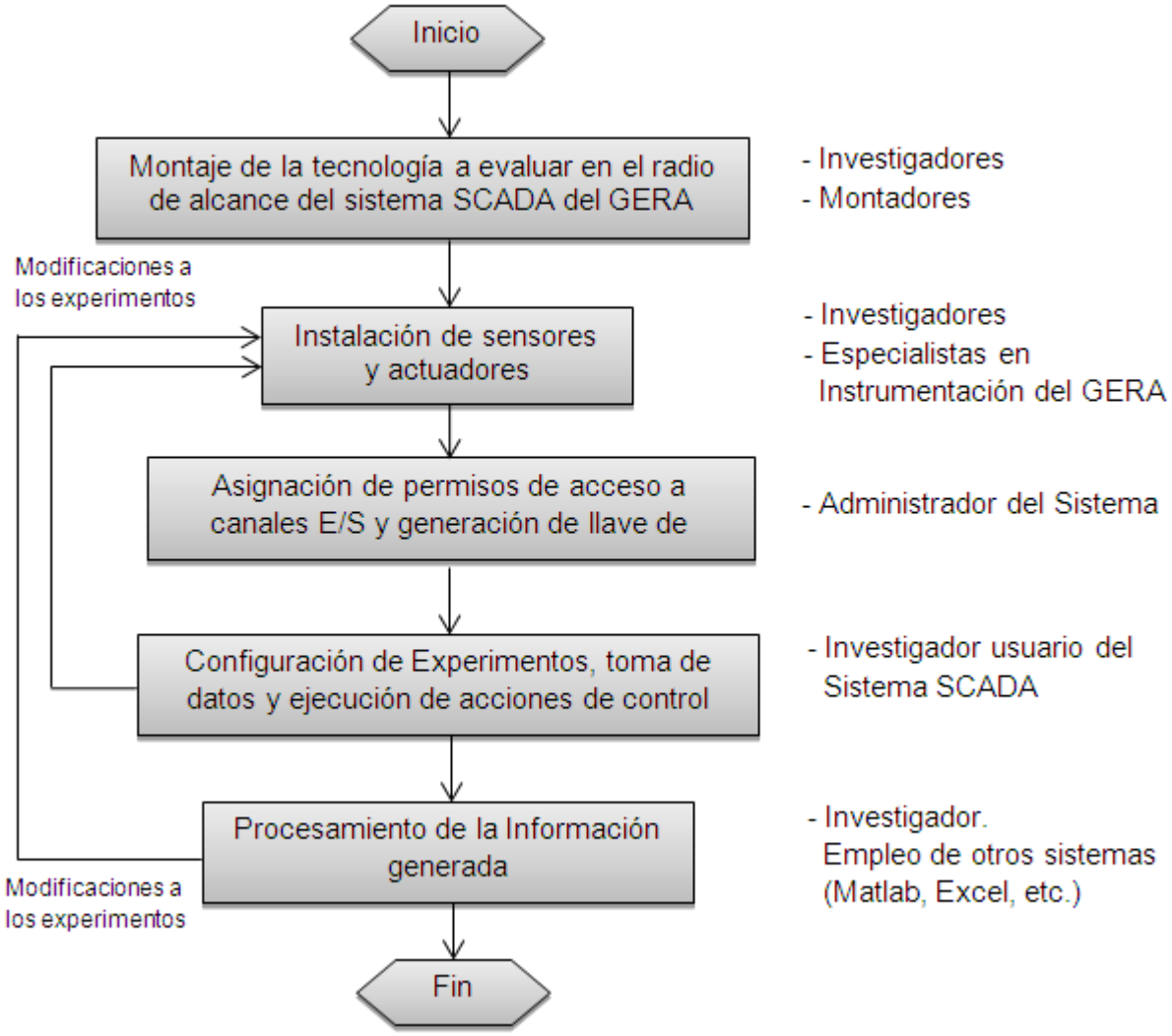


Figura 1.1: Diagrama de flujo del proceso de ensayos y pruebas de laboratorio del GERA

1.1.2 Actual sistema instalado

Debido a la necesidad del centro de desarrollar experimentos y pruebas de laboratorio para adquirir mediciones y ejecutar algunas acciones de control, se dieron los primeros pasos para la conformación de un sistema de adquisición de datos que permitiera la posibilidad de instalar de manera fácil y flexible todo tipo de sensores/transductores y actuadores que cumplieran con los estándares industriales de tensión y corriente (4-20mA, 0-10V, Salidas a Relé, etc.), de igual manera el sistema debía ofrecer la posibilidad de adecuarse a las condiciones físicas del terreno (movilidad). Teniendo en cuenta además que se trataba de un

laboratorio de evaluación de tecnologías de energías renovables se hacía indispensable la necesidad de incluir la toma de datos de al menos una estación meteorológica.

Basados en estas premisas se optó por las tecnologías descritas en los epígrafes subsiguientes para conformar un sistema de adquisición y control modular y remoto, que incluyera la posibilidad de ser comandado mediante un protocolo abierto permitiendo el futuro desarrollo de aplicaciones para la adquisición de datos a la medida. Se decidió además que la estación meteorológica que conformaría el sistema tendría que soportar comunicación activa con una PC (D'Armas, 2012).

El esquema de comunicaciones del sistema se muestra en la Figura 1.2, donde cabe destacar que todos los dispositivos de entradas, salidas y la estación meteorológica fueron ubicados en el mismo bus, donde coexistían diferentes protocolos de comunicación sobre la misma norma física.

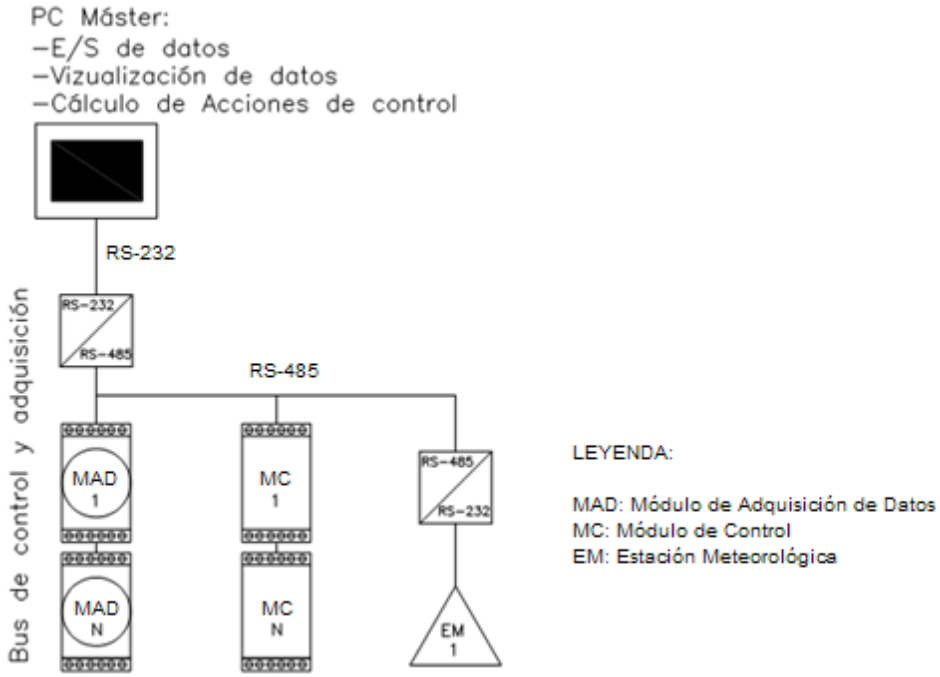


Figura 1.2: Diagrama de comunicaciones del primer sistema SCADA concebido para el GERA.

1.1.2.1 Módulos de adquisición de Datos

Luego de una revisión a los principales sistemas distribuidos que ofrecieran prestaciones enfocadas a la automatización de laboratorios, pruebas a productos y adquisición de datos

remotos, fueron seleccionados módulos fabricados por las compañías ICP DAS Co., Ltd. y Advantech Co., Ltd. y Adlinktech Inc., tomando como base que todos emplean el mismo protocolo abierto de comunicación DCON (ICP DAS Co., Ltd., 2009) sobre RS-485 (Telecommunications Industry Association, 2005) y facilitan la construcción de un sistema expandible con topología en forma de bus.

Estos fabricantes son líderes globales en el diseño y fabricación de equipamiento para adquisición de datos y módulos de Entradas y Salidas distribuidas, ofreciendo dispositivos completamente independientes los cuales son ampliamente utilizados en aplicaciones industriales tales como el monitoreo de instalaciones, monitoreo ambiental y procesos de control. Los módulos de E/S distribuidas están divididos en E/S Analógicas y módulos de E/S Digitales (D'Armas, 2012).

Las principales características de esta familia de dispositivos se listan en la Tabla 1.1 y en el Anexo # 1 se muestra una ficha de los principales dispositivos disponibles en el GERA con los que se han ejecutado los trabajos de desarrollo y validación de comunicación del sistema.

Tabla 1.1: Principales características de los Módulos de Adquisición de Datos

Parámetros	**Valores**
Tipos de entradas	Termopares, Termoresistencias, mV, V, mA
Velocidad de Muestreo	20 muestras/segundo
Exactitud	±0.05%
Desviación de cero	±6 µV/^{o}C
Rechazo a modo común	90 dB
Aislamiento	3000 VDC
Alimentación	10-30 VDC
Consumo	1.2 W
Comunicación	Protocolo DCON sobre RS-485

En resumen estas series de dispositivos son módulos de interfaz sensor-computadora basados en microprocesadores. Pueden ser remotamente controlados mediante un simple set de comandos (Anexo # 2) en formato ASCII y transmitidos mediante el protocolo RS-485. Estos módulos incluyen acondicionamiento de señales, aislamiento, selección de rangos y escalas, conversión A/D y comunicación digital.

1.1.2.2 Módulos de Salidas

Estos dispositivos son básicamente módulos distribuidos de salidas digitales y analógicas con la posibilidad de comunicación mediante la norma física RS-485 y el protocolo Modbus (Anexo # 3). Cada equipo consta de doce salidas agrupadas en cuatro tipos diferentes. A continuación se muestran las principales características de cada uno de estos grupos.

Salidas a Relé:

Parámetros	**Valores**
Cantidad de salidas a relé	5
Carga nominal	5A@250VAC, 5A@30VDC
Voltaje máximo de conmutación	250VAC, 150VDC

Salidas a -10~10 VDC:

Parámetros	**Valores**
Cantidad de Salidas	3
Resolución Máxima	5 mV
Exactitud	±0.5% [Escala completa]
Velocidad máxima de conversión	8 ms/3 Canales
Aislamiento	Optocoplado

Salidas a 4-20 mA:

Parámetros	**Valores**
Cantidad de Salidas	3
Resolución Máxima	4 µA
Exactitud	±0.5% [Escala completa]
Velocidad máxima de conversión	8 ms/3 Canales
Aislamiento	Optocoplado

Salidas Variador de Velocidad:

Parámetros	**Valores**
Cantidad de Salidas	1
Potencia	0.4 kW
Entrada	200 V – 230V Monofásico 50/60 Hz @ 5.5A
Salida	0 ~ Voltaje de entrada @ 0.1 ~ 200 Hz

1.1.2.3 Estaciones Meteorológicas

El sistema que actualmente se encuentra en funcionamiento tiene disponible la posibilidad de comunicación con estaciones meteorológicas de la marca Thies-Clima (D'Armas, 2012) mediante el empleo de una librería dinámica (DLL) que actúa como driver de enlace entre el software y el equipo de adquisición de datos, ya que la trama del protocolo de comunicación serie que emplean estos dispositivos no es de carácter público.

El Datalogger empleado es el modelo DLx-MET considerado por si solo como un sistema completo de adquisición y almacenamiento de datos meteorológicos (Ej. Velocidad y dirección del viento, temperatura, precipitaciones, humedad relativa, radicación solar, etc.) (Thies Clima, 2014)

El instrumento puede ser operado mediante la interfaz de comunicaciones serie RS-485 o mediante el teclado incorporado con Pantalla LCD. Los datos técnicos se muestran en el Anexo # 4.

1.1.2.4 Módulos de Comunicación

En el actual sistema instalado se emplean tres tipos de dispositivos de comunicación que desarrollan las funciones de conversores de medios, por lo que desde la óptica de la capa de aplicaciones del modelo OSI (Tommila, Juhani, & Lauri, 2005) son considerados transparentes ya que estos efectúan el enlace en capas inferiores de la estructura. Estos dispositivos son utilizados para dar soporte a las comunicaciones del sistema en tres casos típicos que son: mediante un BUS cableado, mediante enlaces inalámbricos de corto alcance y mediante enlaces inalámbricos de largo alcance.

Los dispositivos empleados son:

- Pasarela Serie/Ethernet[1]

Parámetros	**Valores**
Marca y Modelo	Hundure, e-P132
Interfaces de Comunicación	LAN: Conector RJ-45, COM; RS-232/RS-485/RS-422
Velocidad Serie	300 bps ~ 115.2 Kbps-N-8-1
Consumo	4.5 W
Enlace	Puerto serie virtual (Ref. Anexo # 5)

[1] Para enlaces cableados mediantes BUS a 2 o 4 hilos

- Pasarela Serie/Wi-Fi[2]

Parámetros	**Valores**
Marca y Modelo	Hundure, eNET-132W
Interfaces de Comunicación	Wi-Fi: IEEE 802.11b/g, COM; RS-232/RS-485/RS-422
Velocidad Serie	300 bps ~ 115.2 Kbps-N-8-1
Velocidad Wi-Fi	54 Mbps
Seguridad	WEP, AES, WPA2
Alcance	Hasta 300 metros (@ 12Mbps en áreas abiertas)
Consumo	5 W
Enlace	Puerto serie virtual (Ref. Anexo # 5)

- Pasarela Serie/RF[3]

Parámetros	**Valores**
Marca y Modelo	ICP DAS, SST-2450
Interfaces de Comunicación	RF: 2.4GHz, COM; RS-232/RS-485/RS-422
Velocidad Serie	300 bps ~ 115.2 Kbps-N-8-1
Alcance	Hasta 1 Km con antena externa opcional de 8 dBi, 15 dBi, 21dBi
Consumo	2 W
Enlace	Puerto serie virtual (Ref. Anexo # 5)

[2] Para enlaces inalámbricos de corto alcance (hasta 300 M)
[3] Para enlaces inalámbricos de largo alcance (hasta 1 Km sin repetidores)

1.1.3 Requerimientos y selección de tecnologías para el nuevo sistema SCADA

Para el análisis de requerimientos necesarios en el nuevo sistema SCADA se tomó como base los epígrafes 1.1.1 y 1.1.2 donde se describe el procedimiento para ensayos de laboratorio y se exponen las principales características de la tecnología instalada respectivamente. Esto sumado al criterio del personal especializado del centro permitió obtener un listado de las principales características no deseadas o perfectibles en el actual sistema, las cuales se listan a continuación:

- Retardos en la toma de datos
- Inestabilidad
- Rigidez en los mecanismos de rediseño y configuración del sistema
- Escalabilidad y adaptabilidad reducida
- Demanda excesiva de recursos de cómputo
- No existencia de mecanismo de autenticación multiusuario

Estas y otras características generan la necesidad de disponer de un nuevo sistema que se adapte mejor a los requerimientos que se demandan en centros de investigación, por lo que para el desarrollo de la nueva aplicación se seleccionarán las herramientas y tecnologías adecuadas abordando los siguientes puntos:

Selección de sensores:

El caso de la selección de sensores es un tema realmente amplio ya que en la actualidad existen múltiples ofertas en el mercado que permiten medir todo tipo de fenómenos naturales. Teniendo en cuenta que el objetivo de este trabajo es obtener un soporte flexible de hardware y software para la adquisición y control de datos que incluya la posibilidad de ser funcional con cualquier tipo de sensor/trasmisor con salida estándar, es válido enfatizar que según el esquema de funcionamiento mostrado en la Figura 1.1 la selección de elementos sensores será competencia exclusiva de los investigadores y especialistas en instrumentación del centro, encontrándose así este tópico fuera del alcance planificado para el desarrollo de esta investigación. No obstante el sistema permite la posibilidad de asociar cualquier indicador en pantalla con algún sensor o dispositivo físico en el campo.

Selección del hardware para la E/S de datos:

Para el desarrollo de este punto se tomarán como premisa los tipos de señales que será necesario medir o generar, la necesidad o no de incluir dispositivos para el acondicionamiento de señales, la velocidad necesaria para adquirir o generar muestras, el menor cambio en las señales que será necesario detectar y el máximo error permisible para la aplicación.

En el caso de las señales a medir o generar será necesario emplear dispositivos que soporten entradas y salidas analógicas, digitales y contadores/temporizadores para registrar eventos digitales o generar pulsos o señales de este tipo. Por lo que se optará por una plataforma modular que pueda ser ajustada a la medida de los requerimientos demandados, posibilitando así la mayor cantidad de configuraciones posibles con un mejor grado de exactitud que dispositivos combinados multifunción. Garantizando de esta manera la posibilidad de expansiones y reconfiguraciones futuras.

Para el acondicionamiento de las señales emitidas por los sensores será necesario que los equipos de medición seleccionados integren todo el proceso de amplificación o atenuación, aislamiento, filtrado, excitación, linealización, etc., sin necesidad de incluir dispositivos externos para la ejecución de estos procedimientos.

Para la selección de la velocidad de muestreo se parte del hecho de que el sistema deseado estará enfocado en mayor medida a la supervisión y control de procesos asociados a las energías renovables y parámetros ambientales como son la temperatura y humedad relativa por lo que se considerará ideal dispositivos que permitan tomar hasta 10 muestras por segundo (National Instruments, 2011) con una exactitud deseada de al menos ±0.5 %.

Selección del bus de comunicaciones:

En el mercado existen centenares de dispositivos de adquisición de datos disponibles con una amplia variedad de buses de comunicación, por lo que se suele hacer dificultoso seleccionar el bus adecuado para las necesidades de la aplicación que se desean. Cada bus tiene diferentes ventajas y respuestas en términos de rendimiento, latencia, portabilidad, distancia desde el "*host*". Por lo que las consideraciones técnicas que se tendrán presentes para la seleccionar el bus de datos serán el volumen de datos que fluirá a través de este, el

grado de portabilidad necesario en el sistema y la distancia máxima permitida entre sistema de medición y la Pasarela o PC.

En el caso del volumen de datos que fluirá a través de bus o ancho de banda será calculado mediante la siguiente ecuación (National Instruments, 2011):

$$Abm = BpM * Vm * C$$

Dónde:

Abm – Ancho de banda mínimo requerido

BpM – Cantidad de Bytes por muestra (se utilizará un valor igual a 4 como media estimada)

Vm – Velocidad de Muestreo (en el sistema se demanda al menos 1 muestra por segundo)

C – Cantidad de canales necesarios (Se requieren 250 canales de medición disponibles)

Obteniendo que $Abm = 1\ MB/s$ por lo que una solución basada en un bus de comunicación serie cableado cumplirá los requisitos planteados. Por otra parte además de esta solución será necesario incorporar mecanismos de enlaces inalámbricos que permitan una rápida y flexible instalación de los dispositivos dentro o fuera del polígono de pruebas.

Selección de la computadora para el procesamiento:

Para la selección de la computadora encargada de efectuar la toma y almacenamiento de datos se demandará que disponga de características adecuadas para el procesamiento paralelo de múltiples hilos de programa, comunicación Ethernet Gigabit y tecnología de almacenamiento RAIL 1 o superior que garantice el adecuado resguardo de la información obtenida. Por lo que se puede concluir que para la instalación del sistema es adecuado un servidor profesional de gama baja tal como los ofertados por reconocidos fabricantes mundiales como DELL, HP o INSPUR.

Selección de la aplicación de software y técnicas de visualización:

Actualmente en el mercado existen decenas de software SCADA con potentes herramientas prediseñadas para la cumplimentación de la mayoría de las tareas que industrialmente se demandan a estos sistemas, desde el enlace y la toma de datos de equipos de disímiles fabricantes, hasta el manejo de alarmas y generación de reportes, pero teniendo en cuenta que el sistema en cuestión requiere características particulares que lo diferencian, se hace inviable el empleo de este tipo de software para la aplicación que se pretende obtener, por lo que se ha optado por otras interfaces de desarrollo de aplicaciones de software industriales mejor enfocadas al campo de la instrumentación virtual tales como LabView o Labwindows/CVI, ajustándose mejor esta última al desarrollo del proyecto, ya que a pesar de que la generación de código en lenguaje ANSI C puede ser un poco más compleja que la solución de bloques de funciones que ofrece Labview, ofrece la posibilidad de una mayor flexibilidad y ajuste a las funciones a implementar.

Esta aplicación será analizada posteriormente como una herramienta adecuada para el desarrollo de sistemas que requieran un mayor grado de especificidades que los clásicos industriales.

Finalmente expuestos los requerimientos para el nuevo sistema, se concluye que no se trata de un SCADA clásico, sino de un sistema SCADA flexible, donde se consideran otros aspectos (como los relacionados a seguir) que por sus características serán tratados en el epígrafe siguiente:

- Selección del formato de almacenamiento de datos
- Selección de herramientas de análisis de información y generación de reportes

1.2 Sistemas SCADA flexibles

Los sistemas de supervisión y control, generalmente reconocidos en la literatura científico-técnica por su acrónimo en inglés SCADA, han sido definidos como la tecnología que posibilita al usuario recoger datos de una o más instalaciones distantes y/o enviar instrucciones de control limitadas a estas instalaciones (Krutz, 2006). Esta definición establece un sistema SCADA como la tecnología que comprende todos los elementos de

hardware y software que intervienen en el proceso de supervisión y control de la planta. Abarcando tanto los controladores, las redes de datos, los Servidores de Datos, los Servidores de Red, los Terminales de Usuarios hasta los protocolos y drivers de comunicación y las aplicaciones de supervisión y control.

Típicamente en un sistema SCADA tradicional el flujo de datos es limitado a destinos predefinidos que están basados en enlaces fijos entre diferentes dispositivos de hardware (Patel, 2004). Sin embargo es importante hacer notar que los requerimientos de este tipo de sistemas en campos especializados como son los laboratorios de investigación y adquisición de datos son mucho más sofisticados y claramente diferentes a los tradicionales sistemas donde las configuraciones estáticas y el flujo de información por canales predeterminados marcan la naturaleza de los mismos. Por lo que para dar solución a la demanda de un sistema SCADA flexible para el Laboratorio del GERA se analizarán las principales características desde tres ópticas diferentes:

Personal desarrollador del sistema:

Desde el punto de vista del personal desarrollador se requerirá que el sistema sea programado modularmente o por bloques funcionales para garantizar la escalabilidad y futuros desarrollos. Debido a la necesidad de ejecución de múltiples tareas de manera concurrente será imprescindible además el empleo de los mecanismos adecuados para el modelado y sincronización de los múltiples hilos de programa así como la correspondiente traducción de los modelos al lenguaje de programación en que se implementará el sistema.

Administrador del sistema:

Desde este punto de vista el sistema deberá poder adaptarse fácilmente y sin necesidad de modificaciones en su estructura a las necesidades planteadas por el cliente y la instalación, Ej. Detección automática de nuevos dispositivos de control y/o adquisición de datos, reconfiguración automática de las prioridades con que se muestrean los canales para garantizar la toma de datos en tiempo real, asignación de permisos de lectura y/o escritura en canales a cada usuario por independiente.

Usuarios del sistema:

Desde el enfoque del usuario final del sistema, este deberá disponer de la posibilidad de crear, guardar y cargar sus propias pantallas, asociar cada elemento en pantalla al origen de datos que estime conveniente, fijar los tiempos de muestreo con que se tomarán datos, seleccionar donde se almacenará la información adquirida, crear y ejecutar reglas y lazos de control de manera gráfica e intuitiva, así como configurar y manejar alarmas y avisos. Lo antes tratado puede ser resumido en la siguiente tabla:

Tabla 1.2: Comparativa entre un sistema SCADA clásico y la nueva propuesta de sistema SCADA flexible.

Aspectos	**SCADA Clásico**	**SCADA Flexible**
Usuarios	Son creados y configurados por el desarrollador del sistema. Los permisos asignados no pueden ser modificados mientras se ejecuta el SCADA.	Se sustituye el concepto de usuarios por llaves de software que asignan permisos a las diferentes copias del software cliente que se ejecutan en la red; los permisos pueden ser modificados por el administrador del sistema en tiempo de ejecución.
Drivers de comunicación	Permiten el enlace de las diferentes variables de la aplicación con direcciones de memorias ubicadas en equipos remotos. El manejo de estos es exclusivo de los desarrolladores del sistema.	Permiten el enlace de las direcciones de memorias ubicadas en equipos remotos con los servidores OPC que componen la aplicación (OPC Foundation, 2014). El manejo de estos se hace de manera automática y totalmente transparente para los usuarios.

Aspectos	**SCADA Clásico**	**SCADA Flexible**
Sinópticos	Son creados por los desarrolladores, donde se representan en pantalla los indicadores gráficos deseados.	Son creados por los usuarios, donde se representan en pantalla los indicadores gráficos deseados. Facilitando así que estos visualicen la información que realmente necesitan.
Almacenamiento de datos	Se efectúa en múltiples formatos de bases de datos pero como tablas aisladas sin relación entre sí.	Se efectúa en múltiples formatos de bases de datos relacionales para facilitar la generación de reportes, el usuario puede además obtener la información en ficheros de texto.
Alarmas	Son configuradas por los administradores del sistema y administradas por los usuarios.	Son configuradas y administradas por los usuarios del sistema.
Comandos ante eventos	Permiten la ejecución de acciones básicas como respuestas a determinadas condiciones previstas. Estos comandos son configurados solo por el personal desarrollador del sistema.	Además de la posibilidad de configurar acciones de control básicas de tipo condición-acción, ofrecen la posibilidad a los usuarios de crear lazos de control en diferentes configuraciones basados en controladores tipo PID, ON-OFF, etc. Sin necesidad de intervención del personal desarrollador del sistema.

Aspectos	SCADA Clásico	SCADA Flexible
Reportes	Por lo general se emplean sistemas externos para la confección y visualización de reportes, tales como Crystal Report, Quick Report, etc. Teniendo la desventaja que el origen de datos a emplear para generar estos informes son tablas acumulativas y no bases de datos relacionales.	Al almacenar la información asociada a cada proceso o experimento supervisado o controlado en bases de datos relacionales y estandarizadas facilita a los sistemas de generación de reportes la obtención de la información requerida sin invertir grandes esfuerzos en el formateo y preparación de datos.
Programación	Permiten la programación empleando lenguajes propios de los PLC como son IL o LD. Soportan además confección de código en lenguajes como Visual Basic Script, Java Script, etc.	Permiten la programación en lenguajes como ANSI C (Ej. LabWindows/CVI), bloques funcionales (Ej. LabView), etc.

1.2.1 El ambiente de desarrollo LabWindows/CVI

Tal como se explicó anteriormente, para el desarrollo de aplicaciones SCADAS Flexibles se hace imprescindible el empleo de ambientes de desarrollos especializados. Tal es el caso del LabWindows/CVI, el cual fue seleccionado como herramienta para dar solución a la implementación del sistema deseado. Este ambiente de desarrollo basado en el lenguaje ANSI C es avalado por más de 25 años en el mercado de la creación de aplicaciones de alto desempeño en el campo de la manufactura flexible, militar, aeroespacial, telecomunicaciones y automovilística.

Además de las potentes librerías propias del lenguaje ANSI C dispone de un gran volumen de funciones e instrumentos enfocados al campo de la instrumentación virtual, control de procesos, procesamiento digital de señales, conexión con bases de datos, etc.

Este ambiente de desarrollo dispone en especial de cuatro características que lo hacen ideal para el desarrollo de una aplicación SCADA Flexible; estas son:

- Generación de código que puede ejecutarse hasta un 60% más rápido que los de otros compiladores en el mercado al estar basado en el estándar industrial para la optimización de la compilación (LLVM).
- Permite el uso flexible, portable y escalable del estándar OpenMP para la programación multiproceso de memoria compartida en múltiples plataformas que permite añadir concurrencia a los programas escritos en C, C++ y Fortran, lo que facilita en gran medida la implementación de aplicaciones previamente modeladas y validadas con sistemas como las Redes de Petri. (En el siguiente epígrafe se tratará en detalles el tema de la programación concurrente).
- Intercambio eficiente de datos entre diferentes aplicaciones en red que garantiza altas tasas de transferencia y latencias comparables al protocolo TCP, incorpora mecanismos que permiten el restablecimiento de la comunicación de manera automática en caso que falle algún dispositivo. Emplea un buffer que garantiza que la transferencia de información siempre se ejecute de manera satisfactoria incluso en redes con conectividad intermitente.
- Empleo de funciones TDMS (Technical Data Management Streaming) para implementar la escritura de datos con alto rendimiento en el disco duro, facilitando la lectura y escritura de estos de manera asincrónica para aumentar el desempeño mediante la pre-localización de los espacios en disco que se van a escribir, lo que garantiza que la información no se copie de manera fragmentada y se optimice la escritura.

1.3 Programación concurrente

Al programar concurrentemente y por ello compartir recursos surgen algunos problemas que necesitan ser resueltos para así aprovechar al máximo todas las ventajas que esta nos puede brindar. Entre los más importantes podemos mencionar algunos:

- Datos Compartidos: La ejecución de un proceso que pueda afectar la información perteneciente a otro proceso que se ejecuta en paralelo a menos que esté autorizado a hacerlo.
- Abrazo mortal: Estado en el que dos transacciones se queden bloqueadas cada una esperando por recursos que está utilizando la otra.
- Inanición: Estado al que llega una transacción cuando es seleccionada repetidamente para abortar y así evitar un abrazo mortal.
- Livelock: Estado en donde una transacción cambia continuamente de estado en respuesta a cambios en otra transacción mientras la otra hace lo mismo, sin conseguir ningún resultado con ello. (Schiper, 1989).

Existen distintas formas para solucionar estos problemas partiendo de mecanismos de bajo nivel como semáforos que obligan a los procesos a ponerse en cola y así evitar problemas con memoria compartida, mecanismos de envío de mensajes para el mismo propósito y hasta mecanismos de más alto nivel como monitores que proveen ciertas operaciones internas que deben ser llamadas para poder modificar los datos asegurando así el control sobre los mismos. Se puede hablar también de mecanismos de sincronización como el "R*endez Vouz*" donde un proceso no escribe hasta que el otro esté listo para leer. (Barry, 2005).

Estos problemas implícitos en la programación concurrente así como los diferentes mecanismos de solución redundan en la necesidad de empleo de una metodología adecuada para el diseño de estos sistemas.

1.3.1 Metodología de diseño de sistemas concurrentes

Cuando se piensa en utilizar la programación concurrente como herramienta de desarrollo es preciso hablar primero de la metodología de diseño de sistemas concurrentes y en particular la más ampliamente usada define los siguientes pasos: (Flemming, 2005).

- **Partición o Descomposición:**

El problema computacional se descompone en pequeñas tareas que forman las unidades de concurrencia ya sea relacionando su nivel de interacción o simplemente haciendo uso de alguna heurística conservando siempre en mente la necesidad de eliminar redundancia en procesamiento y almacenamiento.

- **Coordinación:**

Esto paso define la incorporación de mecanismos que permitan la comunicación y sincronización de tareas que se pueden realizar usando el paso de mensajes o la memoria compartida. Tratando siempre de garantizar que todas las tareas tengan aproximadamente el mismo número de comunicaciones, que cada tarea se comunique sólo con un pequeño número de vecinos y que estas operaciones de comunicaciones puedan realizarse de forma simultánea**.**

- **Aglomeración o Asignación:**

En este paso, las tareas se agrupan basadas en procesos para optimizar el rendimiento, reducir costes de desarrollo y garantizar la flexibilidad y escalabilidad.

- **Proyección:**

En este último paso los procesos se asignan a los procesadores que haya disponibles de forma que se minimice los costos de comunicación y al mismo tiempo se maximice el uso de esos procesadores, es decir que exista un buen balance.

Para cumplir estas etapas existen muchas herramientas de diseño de sistemas concurrentes pero de ellas las más difundidas son las basadas en Redes de Petri (Dingle, Knottenbelt, & Suto, 2009) que se explican en detalles más adelante.

Para su implementación se pueden utilizar diferentes lenguajes de programación tratados en el siguiente punto.

1.3.2 Lenguajes de programación concurrentes

Dentro de los lenguajes para la programación concurrente resaltan por su especial importancia y prestaciones:

Ada:

Uno de los pocos lenguajes que provee estructuras embebidas para programación concurrente y provee herramientas para diseño de software de seguridad crítica y proyectos grandes que requieran portabilidad y mantenimiento. Por esta razón la mayoría del software para aviación está programado en Ada que también fue el primer lenguaje orientado a objetos aceptado mundialmente. El lenguaje lleva este nombre en honor a Ada Byron que fue el primer programador del que se tiene registros, siendo este una mujer e hija del poeta Lord Byron (Parallel Systems Engineering, 2011).

Occam:

Es un lenguaje de procesamiento paralelo diseñado por un equipo en INMOS en conjunto con el diseño del procesador "*Transputer*" y basado en CSP (Communicating Sequential Processes). Este lenguaje incorpora hilos de ejecución fáciles de usar y un amplio soporte de ambientes multiprocesadores. Este puede ser usado con sistemas de memoria compartida o distribuida y es una buena opción cuando se requiere corrección (Parallel Systems Engineering, 2011).

Como se puede observar los principales lenguajes de programación existentes en el mercado no ofrecen prestaciones enfocadas al campo de la supervisión, control y adquisición de datos, por lo que no se ajustan a los requerimientos necesarios para el desarrollo del nuevo sistema SCADA flexible que se pretende en este trabajo.

Tomando como base que el desarrollo inicial del sistema se hizo sobre Labwindows CVI y que este ambiente de desarrollo dispone de todas las herramientas necesarias para aplicaciones de corte SCADA industrial se decidió continuar el desarrollo del proyecto empleando este ambiente a pesar de que no es un lenguaje de programación enfocado a

aplicaciones de programación concurrente. Por lo que se hace necesario emplear herramientas adicionales de especificación y verificación de sistemas concurrentes.

1.3.3 Herramientas de especificación y verificación de sistemas concurrentes

1.3.3.1 CSP (Communicating Sequential Processes)

Es una teoría matemática para especificar y verificar patrones de comportamiento como abrazos mortales o Livelocks que se dan durante la interacción de objetos concurrentes. Su semántica formal y composicional está completamente ligada con nuestra intuición natural sobre las formas en que las cosas funcionan. Se puede ver el modelo como un grupo de componentes organizados en una capa y comunicándose con otra capa de componentes a través de canales unidireccionales. Este modelo nació debido a la necesidad de encapsular la información de tal manera que esta permanezca correcta, facilitar el diseño y poder detectar fallas antes de que estas ocurran. Entre muchas de las ventajas que este modelo brindan esta su semántica sencilla y por ende su facilidad de aplicar, su kernel tan liviano mejorando así el rendimiento de las máquinas y el que haya software del tipo de FDR que permite verificar sí el modelo esta correcto o no. (Barry, 2005)

El enfoque de sincronización que utiliza CSP es el de "*Rende Vuez*", que no permite que un proceso escriba si al mismo tiempo otro proceso está ejecutando una operación de lectura y viceversa, como estas acciones en teoría se deben realizar en paralelo estas deberían ser no bloqueantes (Parallel Systems Engineering, 2011)

Existen también en el mercado aplicaciones que nos permiten verificar computacionalmente modelos especificados con CSP y una de las más sobresalientes es FDR (Baier & Kaoten, 2008).

FDR (Failures-Divergence Refinement):

Permite la verificación de muchas de las propiedades de sistemas de estados finitos y la investigación de sistemas que no pasan ese tipo de verificaciones. Está basado en la teoría de CSP. Fue desarrollado en la universidad de Oxford. Su método de probar si una propiedad se cumple es el de probar el refinamiento de un sistema de transición que captura la propiedad a través de la máquina candidato. También permite verificar el determinismo de una máquina de estados y esto es usado primordialmente para corroborar propiedades de seguridad (Formal Systems, 2001).

Las herramientas de verificación formal en máquinas de estados finitos son eficientes, pero tienen limitaciones al tratar sistemas de automatización híbridos (variables binarias y continuas discretizadas), no siendo así en el caso de las Redes de Petri extendidas.

1.3.3.2 Redes de Petri

Es un modelo gráfico para describir sistemas concurrentes, se puede ver como un grafo dirigido y bipartido donde las dos clases de vértices se denominan lugares y transiciones, se permiten arcos de interconexión entre los dos elementos de estas redes. Al modelar una red de Petri elemental, los lugares representan condiciones, las transiciones representan eventos y la presencia de por lo menos una ficha en un lugar indica que la condición se cumple. A partir de este modelo elemental se han desarrollado diferentes modelos clásicos, extendidos y de alto nivel para innumerables aplicaciones (Murata, 1989)

Desde un punto de vista informal, en una red de Petri, P es un lugar de entrada para la transición T, si existe un arco dirigido que va desde el lugar P hasta la transmisión T. De igual forma se define un lugar de salida. Si todo lugar de entrada para una transmisión T tiene al menos una ficha, se dirá que T está habilitada (siempre y cuando sean unitarios los pesos de los arcos). Cuando una transición habilitada quita una ficha a cada lugar de entrada y agrega una ficha a cada de salida se llama disparo de la transición habilitada (Baier & Kaoten, 2008).

Una marca M para una red de Petri está viva si al empezar en M es posible disparar cualquier transición dada a través de una sucesión adicional de disparos, sin importar que la sucesión de disparos ya haya sucedido.

Una marca en una red de Petri es acotada si existe un entero positivo N que tiene la propiedad de que en cualquier sucesión de disparo ningún lugar recibe más de N fichas (David & Alla, 1992).

Las ventajas que proporciona la utilización de redes de Petri en el tratamiento de sistemas dinámicos de eventos discretos (SDED) en todas sus fases (análisis, síntesis, modelado e implementación) están sobradamente estudiadas y referenciadas (Xing, Hu, & Chen, 2006), (Baier & Kaoten, 2008), (Sanz, 2013). De igual manera se ha estudiado la capacidad que tienen como estándar de tratamiento de la información. Esta característica es la que hace viable y relativamente sencilla la traducción de un modelo desarrollado en PN a otros lenguajes de modelado de SDED, en concreto a los lenguajes de programación de los PLCs y de los SCADAs. Por tanto, una vez diseñadas las aplicaciones informáticas que realizan tales tareas (Jiménez, 2000), con el mismo esfuerzo que cuesta modelar la automatización con PN se puede emplear esa misma automatización en el SCADA. Concluyendo así que se procederá a modelar el funcionamiento del sistema con redes de Petri. Por lo que se hace necesario hacer un mayor acercamiento a las características de este sistema.

1.4 Las Redes de Petri

Red de Petri (PN) es un **término agrupador**, que en el transcurso del tiempo ha venido a designar un amplio número de modelos de sistemas, procedimientos, técnicas y patrones descriptivos relacionados unos con otros en el sentido de que están todos basados en el mismo principio específico (Teoría de las Redes de Petri) (David & Alla, 1992).

Entre las principales ventajas que ofrecen las Redes de Petri se encuentran:

- Formalismo matemático, que facilita la especificación formal de procesos y sistemas en tiempo discreto, evitando ambigüedades (Zapata, 2007).
- Sintaxis y semántica bien definida.

- Herramienta gráfica para modelar Sistemas a eventos discretos (DES). Lo que permite dar un seguimiento del proceso de automatización en todo el tiempo.
- Provee técnicas de análisis de eventos.
- Permite la construcción sistemática de sistemas a eventos discretos y controladores autómatas.
- Facilita el mantenimiento y modificaciones posteriores en los modelos. Esta metodología permite un fácil entendimiento hasta para quienes no han diseñado el modelo, permitiendo así realizar correcciones y mejoras.
- Garantiza el desempeño del modelo en términos de: No Bloqueos, Ciclicidad, Estabilidad y Alcanzabilidad de estados deseados

Una Red de Petri está conformada por los elementos que a continuación se detallan, estos pueden ser visualizados en la Figura 1.3.

- **Lugares y transiciones**: Representan eventos conectados por arcos. Gráficamente los lugares p se representan por circunferencias y las transiciones t por pequeños segmentos rectilíneos. Un lugar representa un estado en el que puede estar parte del sistema.
- **Arcos**: Un arco une un lugar con una transición, o viceversa e indica la dirección de la secuencia de eventos. Nunca es posible unir un lugar con otro lugar, o una transición con otra transición. Además, n arcos de igual dirección son representados con un solo arco etiquetado con el número n, dicho número se conoce como peso del arco, cuando no existe ninguna etiqueta se asume que el arco es de peso uno.
- **Marca**: Se representa por un punto dentro de la circunferencia que representa al lugar. Las marcas nos permiten visualizar el estado del sistema.
- **Tokens**: Son números naturales o puntos colocados en circunferencias que representan a los lugares. Las transiciones son las responsables de moverlos a través de la red.

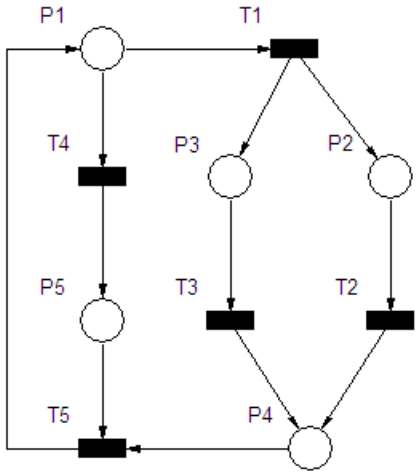


Figura 1.3: Red de Petri

1.4.1 Tipos de Redes Elementales

Existen tres grandes grupos de Redes de Petri: las **redes clásicas**, que incluyen los Sistemas Condición/Evento (C/E-systems), las Redes Lugar/Transición (P/T-nets) y **las extensiones derivadas de ellas**, y las **redes de alto nivel**, que abarcan fundamentalmente las Redes Predicado/Evento (P/E-nets), Redes coloridas y Redes relacionadas. Estos grupos se diferencian fundamentalmente por la forma en que son marcadas las redes (cantidad de marcas o *tokens*) y por la adición de elementos auxiliares.

En los **sistema C/E** sus lugares son simplemente marcados o no marcados (sólo es posible un *token*).

En el modelo de **sistema P/T-nets**, los lugares pueden contener un cierto número de *tokens* sin diferenciación. Esto agrega el análisis de la capacidad de cada lugar (**K**) y el peso asociado a los arcos (**W**).

Las **extensiones a las redes P/T** está dadas por la adición de arcos habilitadores e inhibidores, temporización y otros elementos.

En las **redes de alto nivel**, los lugares son marcados por diferentes tipos de tokens.

Las aplicaciones de automatización con PLCs abarcan principalmente el sector de las Redes P/T extendidas. Un ejemplo clásico es el modelado de la compartición de recursos, como el

uso compartido de los sistemas de comunicación, de áreas de almacenaje para diferentes líneas de producción, etc.

Existen tres reglas de disparo de las transiciones en las P/T-nets:

1. Sensibilización (Habilitación) de transición: donde todos los lugares de entrada de la transición deben contener un número de fichas (Tokens) mayor o igual al peso del arco correspondiente.

2. Ocurrencia del evento asociado con la transición: usualmente el tiempo en las redes extendidas.

3. Disparo de transición: provoca el cambio en el marcaje de la red de M para M', retirando de cada lugar de entrada de t, un número de fichas (Tokens) igual al peso del arco de entrada, y coloca en cada lugar de salida de t, un número de fichas (Tokens) igual al peso del arco de salida.

Las extensiones y modificaciones adecuan el modelo a condiciones específicas de determinadas aplicaciones. Estos son los casos de las Redes Temporizadas (Murata, 1989), Redes Interpretadas (Frey, Analysis of Petri Net based Control Algorithms - Basic Properties, 2000), Redes Extendidas (Silva, Miyagi, Matos, & Afsarmanesh, 1995), y otras más.

La **representación matemática de una PN** viene dada por la definición de la **Matriz de incidencia** que representa la estructura de la red y puede ser descrita de la forma siguiente:

$$A = \ a_{ij}\ _{mxn} \text{ Con } \quad a_{ij} = \ a_{ij}{}^{+} - a_{ij}{}^{-}$$

Dónde:

$a_{ij}{}^{+} = W\ \ j,i$ Es el peso del arco que va de la transición j al lugar i

$a_{ij}{}^{-} = W\ \ j,i$ Es el peso del arco que va del lugar i a la transición j

m Es el número de lugares y

n El número de transiciones.

La Ecuación de estado en una red está dada por:

$$M_{k+1} = M_k + A.V_k$$

Dónde:

M_k Y M_{k+1} son los estados actual y siguiente respectivamente

V_k Es el vector de habilitación actual

1.4.2 Propiedades de las Redes de Petri

En el análisis y diseño de modelos basados en PNs se tienen en cuenta un conjunto de propiedades (Murata, 1989), pudiendo clasificarse éstas como:

- **Propiedades funcionales**: Son aquellas propiedades dependientes del marcaje inicial y reflejan el comportamiento dinámico del sistema:
    - Alcanzabilidad (*Reachability*)
    - Limitación (*Boundedness*)
    - Vivacidad (*Liveness*)
    - Reversibilidad y Estado particular (*Reversibility and Home State*)
    - Cobertura (*Coverability*)
    - Persistencia (*Persistence*)
    - Distancia sincrónica (*Sinchronicdistance*)
    - Disparabilidad (*Fairness*)

- **Propiedades estructurales**: Son aquellas propiedades independientes del marcaje inicial de la red y por tanto inherentes a la estructura de la misma.
    - Vivacidad estructural (*Structural Liveness*)
    - Controlabilidad (*Controllability*)
    - Limitación estructural (*Structural Boundedness*)
    - Conservabilidad (*Conservativeness*)
    - Repetitividad (*Repetitiviness*)
    - Consistencia (*Consistency*)

- Invariantes S y T (*S- and T-Invariants*)
- Disparabilidad limitada estructural (*Structural B-Fairness*)

La definición de cada una de estas propiedades no se encuentra dentro de los objetivos de este trabajo, pero pueden ser encontradas en varios textos básicos sobre PNs como el caso de (Murata, 1989). Por tanto, se trabajará sólo con algunas de ellas por su importancia en el modelado de sistemas de automatización.

En el análisis de modelos PN de sistemas de automatización es muy importante la propiedad de **Alcanzabilidad** de estados, pues a partir de un estado inicial muchas veces se requiere que la red avance automáticamente hasta un estado dado, si no tiene esta capacidad, no podrá ejecutar el comportamiento deseado. A esto está relacionada la **Vivacidad** de la red, si generalizamos esta capacidad a todo el sistema, y es aquí donde pueden detectarse partes de la red que detienen su funcionamiento (como lazos cerrados, bloqueos, *deadloock*), lo cual permite eliminar estas situaciones anormales en el programa a implementar desde esta etapa inicial de diseño.

También los sistemas de control no pueden tener un comportamiento ilimitado en la mayoría de sus elementos por las propias limitaciones físicas del sistema, fundamentalmente capacidad de almacenes o recursos compartidos. Por tanto, también debe vigilarse la propiedad de **Limitación estructural o funcional de la red**.

Reversibilidad y estado particular: La red tiene que garantizar su repetitividad, ya que el trabajo de todo sistema de control es cíclico y más en el caso de los PLC, porque también tiene que garantizar que este comportamiento no sea aleatorio, sino que sea controlable.

Dentro de las propiedades estructurales, las **Invariantes S y T** son importantes medios para el análisis de las redes de Petri, debido a que permiten investigar la estructura de la red, independientemente de los procesos dinámicos.

Las invariantes son **características algebraicas de las redes de Petri** y son usadas en varias situaciones, como para **verificar limitación (boundedness), periodicidad (periodicity) y otras propiedades estructurales.** Su definición formal es la siguiente:

Invariante S para las redes FC: Una solución entera X del sistema homogéneo AX=0, donde A es la matriz de incidencia, donde el conjunto de lugares de dicha P/T-net no tienen cambios del total de Tokens durante los disparos de las transiciones. Estas invariantes se representan por un **vector n-columna x**, donde n es el número de lugares de la red de Petri. Las **entradas del vector x que no son cero, son los lugares invariantes.**

Invariante T para las redes FC: Una solución entera X del sistema homogéneo A^t X=0, donde A^t es la transpuesta de la matriz de incidencia. Indican cómo, comenzando desde alguna marcación M, cada transición tiene que disparar (exactamente v (t) veces), para reproducir este marcación. Estas invariantes son representadas por un **vector m-columna y**, donde m es el número de transiciones de la red. Este vector y **almacena un número entero en las posiciones que pertenecen a las transiciones invariantes**, y almacena el valor cero en los otros casos. Los números enteros representan la cantidad de veces que la transición correspondiente puede ser dispara para que la red vuelva a su estado inicial.

1.4.3 Métodos de análisis de propiedades

Los métodos de análisis de las PNs se pueden clasificar en tres grupos (Murata, 1989):

- Método del árbol de cobertura o de Alcanzabilidad (*Coverabilityor Reachability Tree Method*).
- Método analítico o enfoque por la ecuación matricial (*Matrix-Equation Approach*).
- Método de técnicas de reducción o descomposición.

El primer método incluye la enumeración de todos los marcajes alcanzables. Se puede aplicar a todas las redes, pero está limitado a redes “pequeñas” por su complejidad debido a la explosión de estados para sistemas grandes (inmenso número de lugares).

Los otros dos métodos son prácticos y potentes, pero en muchos casos, solo son aplicables a subclases especiales de PNs o situaciones especiales (tiene limitaciones). Para el análisis se deben asumir, en primer lugar, sólo PNs puras (sin lazos infinitos) o transformadas a puras (agregando un par lugar-transición a cada lazo infinito) para admitir la definición de las ecuaciones matriciales y sus invariantes S y T (Murata, 1989).

El de técnicas de reducción o descomposición facilita y simplifica el análisis de sistemas "grandes" al reducir estructuralmente el modelo obtenido a una descripción más general (Clarke & Wing, 1996), denominada **subred o "macro"**, la cual mantiene las propiedades originales de la red que le dio origen (Murata, 1989). Estas técnicas de transformación para las PNs, se utilizan para analizar vivacidad, seguridad, y limitación mediante la comprobación de buena conformación de la red. Las transformaciones clásicas definidas por (Murata, 1989) son las representadas en la Figura 1.4.

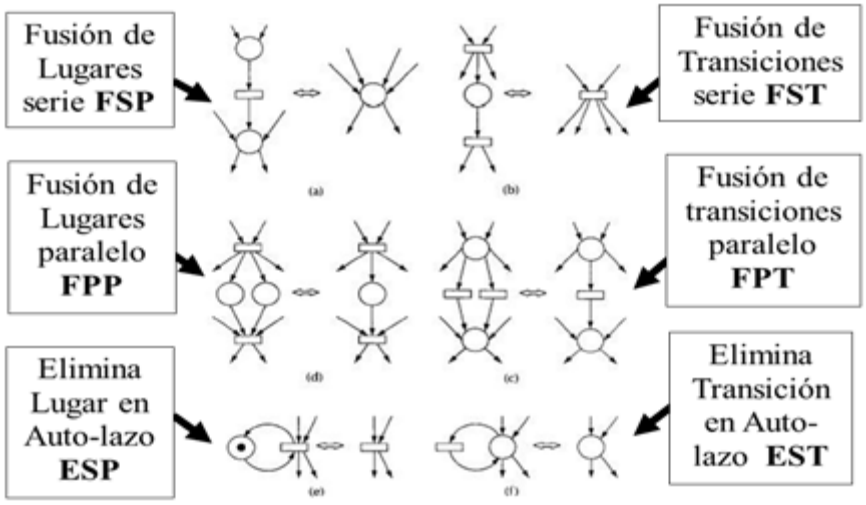


Figura 1.4: Transformaciones básicas en las Redes de Petri

1.4.4 Modelado de sistemas de automatización mediante Redes de Petri

Dentro de las PNs aplicadas al ámbito de automatización se han desarrollado diferentes variantes y extensiones que difieren en su concepción, forma de representación, interpretación y los métodos de análisis que utilizan. (Holloway & Krogh, 1997) Define dos líneas principales hacia las que se orientan estos trabajos:

- **Control de realimentación de estados** (State feedback control): Basado en la adición de lugares de control a la PN, creando las **CtlPN** (Controlled PN).
- **Control de realimentación de eventos** (Event feedback control): Basado en la asignación de eventos a las transiciones, creando las **LabPN** (Labeled PN).

Las **CtlPN** son una tripleta $N_c = N, C, B$, donde $N = P, T, F$ es una red de Petri, C es un conjunto finito de lugares de control disjuntos de P y T, mientras que $B \subseteq CxT$ es el conjunto de arcos desde los lugares de control a las transiciones.

Las **LabPN** (*Labeled* PN o PN *generator*) son una quíntupla $G = N, \Sigma, \mathrm{l}, M_0, \mathrm{F}$, donde $N = P, T, F$ sigue siendo una PN, Σ es un conjunto (alfabeto) finito de eventos, $\mathrm{l}: \Sigma \rightarrow \mathrm{T}$ es una función que asigna eventos a las transiciones, M_0 es el marcaje inicial y F un conjunto finito de marcajes finales.

Sobre la primera tendencia se han desarrollado recientemente las **Automation Petri Nets** (APN) (Uzam, 1998) y **GHENeSys** (Glez & Silva, 2001), mientras que a la segunda se asocian las ***Signals Interpreted Petri Nets*** (SIPN) (Frey, Analysis of Petri Net based Control Algorithms - Basic Properties, 2000).

En la primera tendencia fue desarrollado el modelo **APN (Autómata PN**) (Uzam, 1998), lo cual está definido como $APN = (P, T, Pre, Post, In, En, X, Q, Mp)$, donde P son los lugares, T son las transiciones, $Pre\ PxT$ *y* $Post\ TxP$ son arcos normales entre P y T, In (arcos inhibidores) y En (arcos habilitadores) lo que representa la conexión de lugares auxiliares asociados a la presencia o ausencia de lectura de sensores o informaciones del sistema, X es el conjunto de condiciones asociado a las transiciones, Q es el conjunto de acciones atribuidas a los lugares (ellos pueden ser señales de Impulso y de nivel), M_p es la marcación inicial.

También se asocia a esta primera línea la Red de Petri **GHENeSys** (Glez & Silva, 2001) que puede ser definida como un séxtuplo $\mathrm{N} = (\mathrm{L}, \mathrm{A}, \mathrm{F}, \mathrm{K}, \mathrm{M}, \mathrm{Q}, \Pi)$ donde: los elementos del conjunto L son llamados lugares y está compuesto por la unión de los conjuntos B y P (boxes y pseudoboxes respectivamente). Los elementos del conjunto A son llamados actividades. F Es la relación de flujo ($\mathrm{F} \subseteq (\mathrm{LxA}) \cup (\mathrm{AxL})$) y sus elementos son llamados arcos. $\boldsymbol{K}: \boldsymbol{B} \rightarrow \boldsymbol{N}+ \cup \{\infty\}$ Es la función de capacidad que indica la capacidad máxima permitida para cada Lugar B.

$\boldsymbol{M}: \boldsymbol{L} \rightarrow \boldsymbol{N}^+$ Es la marcación inicial de la red con respecto a las capacidades de cada lugar, $\boldsymbol{Q}$ es una función que asocia acciones de nivel (atribución de resultado binario) para el subconjunto de los Boxes ($\boldsymbol{B}$) o de impulso (Activación o Desactivación, "*Set* o *Reset*") para

el disparo de algunas actividades ($\boldsymbol{A}$). Π Es una función que marca la diferencia entre los elementos $\boldsymbol{L}$ y $\boldsymbol{A}$, atribuyendo el valor "0" a los lugares simples y "1" a los Macro-elementos (elementos que representan sub-redes)

En esta definición inicial, fue incluida también **l**a función $\boldsymbol{Q}$ (Benitez, Silva, Villafurela, Gomis, & Sudriá, 2008), que permite a GHENeSys estar mucho más cerca de la programación en PLCs, porque vincula su interpretación real con la estructura de la red de PN. Además, esta inclusión permite hacer traducción directa del modelo en PN para el programa en IEC1131.

A la segunda línea pertenecen las **Signal Interpreted PN** (Frey, Analysis of Petri Net based Control Algorithms - Basic Properties, 2000) que está definida como $\boldsymbol{SIPN = (P,T,F, M0,I,O,\varphi,\omega,\Omega)}$ donde $\boldsymbol{(P,T,F,M0)}$ es una PN con su marcación inicial M0, ***I*** es un conjunto de entradas lógicas, ***O*** es el conjunto de las salidas lógicas, φ asocia toda transición a una condición booleana (binaria) de *I*, ω asocia todos los lugares a los valores (0,1, -) de una salida *O,* Ω es una función que combina los valores obtenidos para cada salida con los lugares para evitar conflictos e indeterminaciones..

(Frey, Analysis of Petri Net based Control Algorithms - Basic Properties, 2000) Indica dentro de la definición de SIPN que la función Ω asocia a cada salida condiciones de: indefinidas (-), cero (0), uno (1), contradictorio (c), redundante a cero (r0), redundante a 1 (r1), y combinaciones de c con las dos últimas (c0, c1, c01). Aquí aparece una cuarta regla de disparo que habla de la repetición (prácticamente instantánea) de las sucesiones de disparo hasta que se aparece un estado estable donde ninguna transición es disparada (el proceso de disparo es iterado hasta que una marcación estable es alcanzada y mientras que esto sucede no cambian las condiciones de entrada (todo es incluido dentro de un ciclo del PLC)). Después que la marcación estable sea alcanzada las señales de salidas son recalculadas aplicando Ω a esta marcación. Esto complica el comportamiento de la red que queda lejos de las OPN.

En resumen, la red GHENeSys de (Glez & Silva, 2001)y la red APN de (Uzam, 1998) pertenecen al grupo de **CtlPN** donde hay gran semejanza entre los **Lugares de Control (CtlPN),** las señales de sensores (APN) y los pseudoboxes (GHENeSys), en tanto la SIPN de (Frey, Analysis of Petri Net based Control Algorithms - Basic Properties, 2000) pertenece al grupo de **LabPN donde son atribuidos eventos a las transiciones**.

Los tres modelos (APN, GHENeSys y SIPN) usan la particularidad de **atribuir acciones para el proceso para los lugares** que permiten la extensión al trabajo de desempeño en el proceso controlado que no están presentes en las definiciones iniciales de CtlPN y LabPN de Holloway [31], pero en el caso de las SIPN asocia el valor de la salida y no la acción, lo cual obliga a tener una **función adicional (Ω)** de ajuste del valor final de la salida al terminar cada ciclo del PLC, lo que aumenta la explosión de estados.

Estas líneas también se difieren en la **forma de tratar las señales de los sensores** de estados del proceso. La variante de SIPN (cerca de las LabPN) limita la expresividad gráfica de la PN, al no representar en la estructura del modelo la acción de los sensores, tanto en el gráfico, como en la ecuación de estado. El modelo APN las representa gráficamente, pero también puede utilizarse la atribución de eventos a las transiciones. Mientras que el modelo GHENeSys sólo permite los lugares auxiliares sin eventos de sensores lo que da una representación gráfica más completa.

En resumen, la variante de las **CtlPN** es considerada más esclarecedora y cercana a la teoría clásica de PNs, pero adicionando las facilidades de los **arcos inhibidores y habilitadores** para la conexión de los **Lugares auxiliares a las transiciones** (presente tanto en el caso de las APN como en GHENeSys). La acción de los sensores está de este modo garantizando su inclusión en el modelo con la persistencia de la marcación en esos Lugares especiales (ellos no son afectados por el disparo de transiciones). Las APN incluyen un campo más amplio (ellas pueden usar peso de arcos y hasta diferenciación de tokens) que puede llegar hasta las redes coloreadas, perdiendo así el poder de cálculo matricial simple. Esto obliga a que los métodos de análisis y síntesis con APN estén basados sólo en el análisis de gráficos de alcanzabilidad (RG), influenciado directamente por la explosión de estados. Las redes GHENeSys permiten usar las herramientas de análisis tanto de RG como las matriciales simples.

1.5 Empleo de las redes GHENESYS

Como se comentó anteriormente en un gran campo de aplicaciones es aconsejable el uso de GHENeSys para el modelado de los sistemas de automatización industrial, ya que garantiza

un mayor uso de las herramientas de análisis y síntesis. También las redes GHENeSys están más cerca de los modelos más simples porque ellas permiten que los **lugares auxiliares** (pseudoboxes) no sólo representen mediciones de sensores del proceso, sino también estados de información que vienen de otras sub-redes o de partes de la misma red, lo que simplifica la estructura de redes complejas (Glez & Silva, 2001), y lo que permite mayor efectividad de las herramientas clásicas de análisis y síntesis de PNs para estos controladores.

La definición de **sub-redes (Macro-elementos o simplemente Macro)** en GHENeSys ayuda a la modularidad del modelo, favoreciendo la reusabilidad y la conformación de redes jerárquicas (Figura 1.5). Estos macro-elementos también permiten a GHENeSys abarcar el campo de las aplicaciones no binarias en programación, sin tener que pasar a redes de alto nivel (como las APN) y con una mayor semejanza con los lenguajes de PLCs.

La red GHENeSys permite la creación de módulos en bibliotecas que pueden construir un modelo mayor basado en estructuras típicas de control y entonces sumar una jerarquía. Esto favorece la construcción del modelo y su traducción a lenguajes destinados a la automatización y control industrial.

El modelado es desarrollado de forma modular por medio del uso de los subsistemas y estructuras típicas (Glez & Silva, 2001), (Benitez, Silva, Villafurela, Gomis, & Sudriá, 2008). Por ejemplo, el caso de los motores es modelado en PN para sistemas de automatización como una estructura típica de tres estados: Parado, Funcionando y en Fallo.

La dinámica de las redes es muy importante para el modelado de sistemas de automatización. En GHENeSys son aplicadas las mismas tres reglas de disparo de las PN clásicas, adicionando el caso particular de marcación permanente en los pseudoboxes. En el disparo, las transiciones de macro-elementos se comportan como elementos simples (Normales) pero la interpretación es realizada con una llamada a una sub-red que desarrolla una operación o función preprogramada por un especialista de PLC o por un usuario en un bloque funcional almacenado en la biblioteca del software de edición de programas.

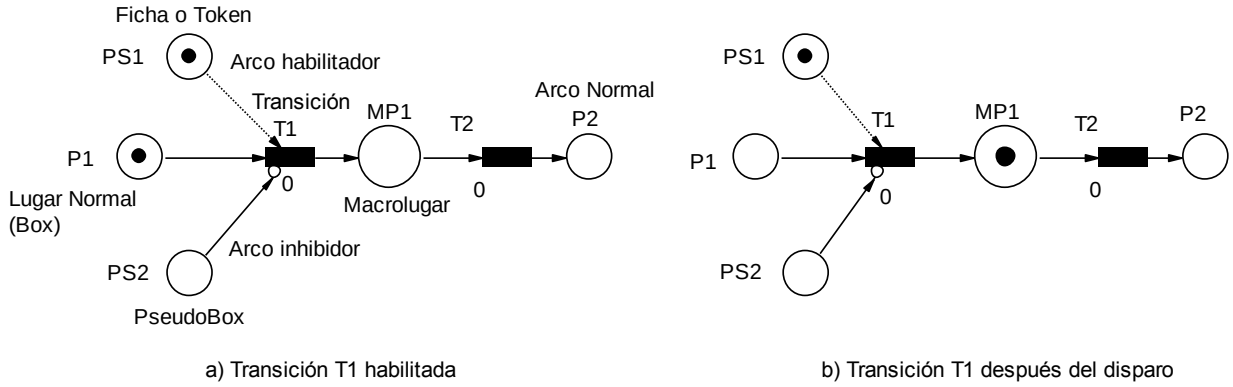


Figura 1.5: Tipos de elementos y reglas de disparo de PN extendidas GHENeSys

A continuación se presenta la metodología propuesta por (Benitez, Silva, Villafurela, Gomis, & Sudriá, 2008) para el proceso de diseño de sistemas de automatización. Se compone por los siguientes pasos:

1. Estudio del sistema a controlar y los requerimientos funcionales del usuario.

2. Determinación de la mayor cantidad de unidades funcionales del sistema de control, sin coincidir necesariamente con su futura implementación.

3. Definición de acciones internas e interdependencias de cada unidad funcional y el nivel jerárquico en que deben estar de acuerdo a estas interdependencias y su función en el sistema de control.

4. Utilización de un diseño Arriba-Abajo (*Top-down)* para establecer un modelo jerárquico del sistema, estableciendo una red GHENeSys para todo el modelo del sistema de control.

5. Refinamiento *Botton-up* del modelo para detallar la estructura de cada subred del modelo, buscando su representación en un tipo de PN lo más sencilla posible. En casos que su simplificación sea imposible, y se requieran redes más complejas, se deben crear nuevas subredes para atender solo a esa situación. Esto permite aplicar a la mayor parte del sistema controlado métodos de diseño de modelos simples de PNs y las herramientas complejas solo en pequeñas subredes.

6. Clasificación de cada subred y determinación de sus propiedades funcionales (fundamentalmente *live* y *safe*) y estructurales (fundamentalmente S- y T-Invariantes).

7. Aplicación del método de reglas de reducción simples para revisar la estructura de la red, y lograr la mayor independencia entre selección y concurrencia de las ramas del modelo.

8. Reclasificación de cada subred modificada y re determinación de sus propiedades, realizando adecuaciones que permitan su cumplimiento.

9. Simulación del trabajo de cada subred y comprobación del cumplimiento de los requisitos funcionales del usuario.

10. Reclasificación de cada subred modificada y re determinación final de sus propiedades.

11. Estudio de las características particulares del equipamiento para su implementación y modelado de la subred que realiza la sincronización de las comunicaciones.

12. Traducción del modelo a un programa SFC en todos los niveles, considerando el criterio del usuario para seleccionar otro tipo de lenguajes (ST, IL, LD) para redes básicas.

Los pasos fundamentales de la metodología para garantizar la efectividad del modelado están concentrados en la verificación de propiedades y en la validación de requisitos funcionales del sistema. Por tanto a continuación se definen estos dos conceptos tan importantes en Ingeniería de Software. El Glosario de IEEE Standard define Validación y Verificación (V&V) como:

“El proceso de determinar si los requerimientos de un sistema o componente están completos y correctos, si los productos de cada fase de desarrollo cumplen los requerimientos o condiciones impuestas por la fase previa, y si el sistema o componente final tiene conformidad con los requisitos especificados.”

Diferentes autores distinguen validación y verificación como dos pasos complementarios de garantía de calidad (Medling, 2009), (Simonak, Hudak, & S., 2009)**,** (Du, Jiang, & Zhou, 2009). En resumen argumentan las dos etapas de la siguiente forma:

Verificación: Está orientada a garantizar las propiedades generales del modelo y la satisfacción de una fórmula dada por el modelo. Su principal característica es que se refiere a la corrección interna de un modelo del proceso. Como opera sobre la estructura formal del modelo del proceso, puede ser realizada sin considerar el mundo real.

Validación: Aborda la consistencia del modelo con el universo de discurso, o sea, el proceso del mundo real. Como es un criterio de corrección externa, es más difícil y ambiguo para decidir. Exige la consulta de las especificaciones y un proceso de discusión con las partes interesadas. A pesar de que validación exige el juzgamiento humano como una característica fundamental, importa resaltar que los métodos formales son útiles para apoyarlo. Por ejemplo, simulación, animación o derivación de declaraciones en lenguaje natural facilitan la validación de un modelo de proceso por parte de los utilizadores.

1.5.1 Verificación sobre GHENeSys

Durante la verificación formal, la garantía inicial de una reducción de errores en el diseño parte desde la buena formación de los modelos creados. A continuación se resumen 7 recomendaciones que deben seguirse en cualquier proceso de modelado (Medling, 2009):

1. Usar la menor cantidad posible de elementos en el modelo.
2. Minimizar las rutas o caminos por elementos.
3. Usar redes con sólo un inicio y un fin.
4. Realizar modelos lo más estructurados posible.
5. Evitar selecciones indeterminadas mediante elementos OR.
6. Usar la mayor cantidad de comentarios e informaciones adicionales a cada paso del modelo.
7. Descomponer los modelos con más de 50 elementos.

El uso de estas reglas es fundamental en el desarrollo de aplicaciones de automatización, ayudando también a la simplicidad y posibilidades de expansión y reparación.

En GHENeSys se aprovecha la propuesta de transformar el modelo en una PN pura (PN sin fuentes y sumideros) (Medling, 2009) al eliminar los pseudoboxes y los arcos que van a las transiciones controlables para de este modo ejecutar la verificación de las propiedades de la red como Vivacidad y Seguridad en una PN pura. De este modo se cumple la condición necesaria de Vivacidad propuesta por (Murata, 1989) (no tener fuentes y sumideros). También, muchos de los modelos de los subsistemas controlados generalmente pueden tener una baja clasificación dentro de las PN clásicas, lo cual permite métodos más simples de verificación.

La mayoría de las aplicaciones de automatización pueden ser consideradas dentro de la gama de las **Redes de Petri de Libre Selección (Free-Choice, FC-nets)** (Murata, 1989). Entonces el análisis de Vivacidad y Seguridad de los modelos controlados propuesta en GHENeSys puede usar los métodos propuestos por (Desel & Esparza, 1995) donde las reglas de reducción dan un método alternativo de análisis de buena-formación (*Well-Formedness*) en FC-nets. El algoritmo de prueba de buena-formación puede ser transformado fácilmente en un algoritmo de prueba de Vivacidad (*Liveness*) y Seguridad o Limitación (*Boundedness*) de sistemas FC.

La verificación de sistemas informáticos requiere de una plataforma para el modelado del sistema y de un método de verificación. Las herramientas de diseño para GHENeSys se encuentran todavía en proceso de desarrollo (Olivera, 2009). Por tanto, en este trabajo se utiliza como plataforma de modelado el Visual Object Net 2.7 (VON2.7) con los modelos de redes de Petri extendidas GHENeSys y se emplean dos métodos de verificación formal: reglas de reducción simples y el enfoque analítico mediante la ecuación de estado. Como el VON2.7 no dispone de opciones automáticas de cálculo analítico de los modelos, se utiliza la herramienta PIPE (Bonet, Llado, Puijaner, & Knottenbelt, 2007).

Se considera al VON2.7 como la herramienta que mejor se adapta a nuestros intereses por sus facilidades visuales de edición y simulación, aunque tiene la desventaja de no dar grandes facilidades de análisis, cálculo de invariantes, técnicas de reducción automática,

implementación automática a lenguajes compatibles con la norma IEC1131, etc. No obstante, éstas pueden ser suplidas parcialmente añadiendo una herramienta como PIPE 4.3, que aunque no cuenta con las facilidades de simulación que VON, si implementa la posibilidad de análisis de propiedades funcionales, de desempeño, de invariantes, extracción de la matriz de incidencia entre otras menos comunes como la clasificación y comparación de PNs y que sólo utilizaremos para obtener las invariantes.

En la industria a menudo es considerado que lo importante es tener una representación detallada del sistema y simularlo. Sin embargo cuando los resultados no se corresponden con lo esperado (cuando contradicen el modelo cognitivo implícito del diseñador), tres conclusiones pueden derivarse:

- El sistema realmente se comporta así (el modelo cognitivo es incorrecto y/o incompleto)
- Hay un error en la especificación del modelo (debe corregirse para ajustarse al modelo cognitivo)
- Hay un error en el simulador (contactar al asesor del programa)

Por tanto, es importante destacar que sin un profundo análisis y verificación, es muy difícil seleccionar una de estas opciones. De aquí que se debe siempre verificar propiedades antes de validar mediante simulación

1.5.2 Validación sobre GHENeSys

Los métodos de análisis de modelos en PN incluyen la confirmación de si existe correspondencia funcional entre el modelo con las especificaciones de exigencias originales (típicamente expresadas informalmente), lo que es probado en la Validación (Girault & Valk, 2001). Alcanzar esta correspondencia requiere de experiencia de modelado y conocimiento de las técnicas que ayudan a la construcción de modelos. Para este objetivo es incluido también el completamiento (*completeness*) de las especificaciones de exigencias. Estas generalmente son expresadas como relaciones de E/S del sistema. Hay entradas que no están definidas en las exigencias iniciales y que deberían ser completadas. Otro aspecto

importante es la Consistencia (*consistency*) de las especificaciones de exigencias. No existe Consistencia (inconsistencia) cuando una combinación de entradas da varias combinaciones de salidas. Todo eso debería ser probado minuciosamente durante la validación del modelo. (Girault & Valk, 2001) Considera también la simulación de eventos discretos como otra variante para la prueba de las propiedades del sistema. Para esto es usado un algoritmo de ejecución que simula el funcionamiento del modelo de la red. Esta es una técnica extensa y consumidora de tiempo. Muestra la presencia de propiedades indeseables, pero no prueba la Corrección (*Correctiness*) del modelo en casos generales. No obstante, permanece como una propuesta para muchos autores. Específicamente se considera esto una herramienta de análisis de comportamiento cuando las limitaciones de memoria de los computadores no permiten generar los gráficos de alcanzabilidad. Se presenta que la simulación permite observar cuantas veces un lugar es marcado o no y calcular la probabilidad de esto. Entonces la simulación de la dinámica de tokens puede generar análisis estadístico importante del comportamiento de la red. Mucho más cuando la herramienta de simulación permite trabajar con tiempos reales de ejecución del sistema en el modelo.

En la validación en GHENeSys de esta metodología debe tenerse en cuenta el estudio simulado del comportamiento del modelo controlado para verificar el cumplimiento de las restricciones de estados prohibidos y la secuencia del comportamiento correcto del controlador.

Considerando el modelado modular jerárquico en GHENeSys, las sub-redes para analizar son propias (una entrada, una salida y caminos de E/S que cubren todos los nodos de la red) y por tanto durante la verificación simulada se debe alcanzar la vivacidad de la sub-red. Entonces como estas sub-redes no tienen dimensiones grandes pueden ser restablecidos los pseudoboxes y realizada la simulación del comportamiento como un medio de evaluación de la ejecución de las especificaciones del usuario para el sistema controlado.

Por ello es muy importante definir una herramienta eficaz en la simulación del comportamiento de una red, que en este caso será el Visual Object Net 2.7. (Drath, Hybrid Object Nets: An Object Oriented Concept for Modeling Complex Hybrid Systems, 1998)

Conclusiones Parciales

- Los sistemas SCADAS Flexibles para laboratorios tienen características especiales por lo que requieren en su proceso de desarrollo el empleo de herramientas de modelado y verificación antes de su implementación.
- Las Redes de Petri Jerárquicas Extendidas GHENeSys son una excelente herramienta de diseño formal para este proceso garantizando la flexibilidad y eficiencia del sistema creado para las particularidades de los SCADA de experimentación en centros de investigación como el GERA.
- La programación concurrente facilita la implantación de aplicaciones escalables y flexibles, debido a que permite modelar el mundo como realmente es. Si bien esta trae consigo algunos problemas, existen mecanismos de control para los mismos y al contrario existe hoy en día una gran cantidad de lenguajes y aplicaciones que facilitan la implantación de aplicaciones usando la programación concurrente y otras que permiten la verificación de la misma.
- El LabWindows, aunque no es un software de programación concurrente tiene grandes facilidades para cumplir los requerimientos de dicha programación dentro de sus excelentes funcionalidades para creación de sistemas SCADA.

CAPITULO II: SISTEMA SCADA FLEXIBLE PARA EL LABORATORIO DEL GERA

Introducción

En este capítulo se diseña el esquema de hardware necesario para el proceso de supervisión y control ajustado a las necesidades y condiciones inherentes de centros de investigación. Finalmente y siguiendo el principio de diseñar, modelar, simular, implementar y validar se desarrolla el sistema de software concurrente que comandará el funcionamiento del hardware instalado.

2.1 Diseño y concepción del nuevo SCADA Flexible del GERA

2.1.1 Ideas conceptuales

El nuevo sistema SCADA para el GERA estará compuesto básicamente por un soporte de hardware distribuido en tres buses de datos y un soporte de software descentralizado en tres aplicaciones independientes que interactúan entre sí mediante una red de datos Ethernet (**Figura 2.1**)

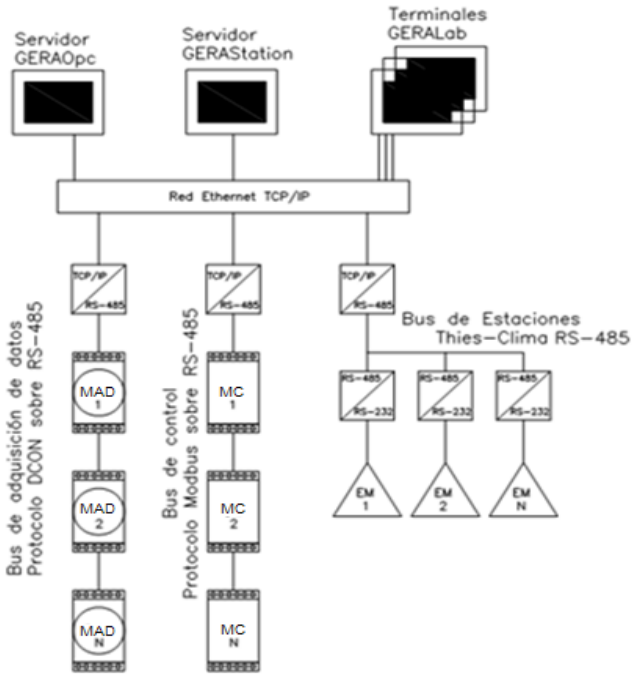


Figura 2.1: Esquema mono lineal de comunicaciones del nuevo sistema SCADA del GERA.

Los tres buses de datos del bloque de hardware serán dispuestos de la siguiente manera:

- **Bus de adquisición de datos**, basado en el protocolo DCON sobre la norma física RS-485. Donde se enlazarán múltiples tarjetas de adquisición de datos que soporten este protocolo (Ref. Epígrafe 1.1.2.1).
- **Bus de control**, sobre la norma física RS-485 donde se instalarán tarjetas propietarias (Ref. Epígrafe 1.1.2.2) con comunicación Modbus y salidas digitales y/o analógicas con niveles de corriente y tensión estándar para el control de múltiples actuadores.
- **Bus de Estaciones Meteorológicas**, basado en el protocolo de comunicación de estaciones meteorológicas Thies-Clima (Ref. Epígrafe 1.1.2.3) sobre la norma física RS-485. Mediante este bus se podrán enlazar al sistema hasta 10 estaciones meteorológicas de este tipo.

De las tres aplicaciones de software que conformarán el nuevo sistema dos de ellas garantizarán el acceso a los buses de comunicación (*GERAOpc* y *GERAStation*) y la tercera aplicación (*GERALab*) será el software central del SCADA y se comunicará con las dos aplicaciones antes mencionadas mediante el estándar OPC sobre el protocolo TCP-IP (OPC Foundation, 2014). A continuación se expone una breve descripción de las características que deberán cumplir estas aplicaciones:

- **GERAOpc,** será un software OPC "*auto-gestionado*" ya que al enlazarse con el bus de adquisición de datos y el bus de control, será capaz de detectar de manera autónoma todos los canales de E/S instalados en estos. Brindando acceso directo al Software GERALab para la lectura o escritura de datos según corresponda. La aplicación será diseñada de manera que se pueda modificar el protocolo de comunicación sin necesidad de efectuar modificaciones estructurales a la misma. Este software será instalado en una PC independiente.

- **GERAStation,** funcionará de manera similar al GERAOpc, pero en este caso brindará soporte de acceso a datos instantáneos adquiridos desde una red de estaciones meteorológicas (de tipo Thies-Clima). La aplicación será diseñada de manera que se pueda modificar el protocolo de comunicación sin necesidad de efectuar

modificaciones estructurales a la misma. Este software será instalado en una PC independiente y almacenará datos históricos en una base de datos MySQL con el propósito de facilitar la visualización de esta información desde sitios web basados en PHP (Welling & Thomson, 2005).

- **GERALab**, este será el software central del SCADA y podrá ser instalado en múltiples estaciones de trabajo enlazadas a la red, se comunicará con los servidores OPC *GERAOpc* y *GERAStation* mediante una red Ethernet.

  Ofrecerá las siguientes posibilidades a los investigadores o usuarios del sistema:
  - Diseño HMI personalizados (Gráficos de tendencias, valores puntuales, etc.)
  - Seleccionar los canales de medición que deseen muestrear.
  - Seleccionar el tiempo de muestreo de los canales de medición.
  - Modificar manual y gráficamente (mediante indicadores en pantalla) las salidas analógicas y/o digitales asociadas al experimento desarrollado por el investigador. (Este punto requerirá una autorización previa del administrador del software GERAOpc)
  - Crear reglas y lazos de control que relacionen entradas y salidas físicas del sistema (Ej. Configurar condiciones del tipo IF…ELSE, lazos de control PID, ON-OFF, etc.)
  - Configurar notificaciones y alarmas
  - Generar un único fichero donde se relacionen en una misma unidad de tiempo los datos tomados de los canales de medición seleccionados, las acciones de control (automáticas o manuales), las alarmas y las notificaciones generadas por el sistema. Esto con el objetivo primordial de los centros de investigaciones, realizar estudios y análisis posteriores del comportamiento y funcionamiento de las tecnologías y sistemas evaluados.

2.1.2 Detalles técnicos ejecutivos para la implementación de sistema

A continuación se realiza un desglose de las consideraciones que se tendrán en cuenta al implementar cada uno de los bloques que compondrán el sistema.

2.1.2.1 Bloque de Hardware

Bus de adquisición de datos:

Como se mencionó anteriormente el soporte físico del sistema estará basado en un bus donde se enlazarán entre sí diferentes módulos de adquisición de datos. En este caso se emplearán dispositivos de la serie I-7000 del fabricante ICP DAS (Ref. Epígrafe 1.1.2.1) y módulos de la serie ADAM 4000 del fabricante Advantech (Ref. Epígrafe 1.1.2.1). Estos dispositivos son básicamente módulos inteligentes que facilitan la interfaz sensor - computadora y están diseñados a base de microprocesadores. Son controlados remotamente mediante el Protocolo DCON (Anexo # 2) sobre la norma física RS-485 (Telecommunications Industry Association, 2005). Incluyen nativamente el acondicionamiento de señales, aislamiento, linealización, conversión A/D y funciones digitales de comunicación. La alimentación eléctrica será mediante el estándar industrial de 24 V aunque los módulos disponibles son funcionales en el rango de +10V a +30V DC.

El bus de adquisición de datos recorrerá perimetralmente el polígono de pruebas del GERA y se canalizará hasta una pasarela RS485/Ethernet ubicada dentro del gabinete de red instalado en el segundo nivel del edificio de este centro. Para esto se empleará cable de control trenzado y apantallado de dos hilos de cobre de 1.5 mm^2 cada uno.

Bus de control:

El bus dedicado a la manipulación de diferentes actuadores estará compuesto por múltiples tarjetas propietarias desarrolladas en el centro de investigación en cuestión y basadas en micro controladores PIC de la serie 18F. Estas son totalmente compatibles con el estándar Modbus RTU (Ref. Epígrafe 1.1.2.2) por lo que quedará disponible la opción de adicionar otros dispositivos que soporten este protocolo (Variadores de Frecuencia, Servomotores, PLC, etc.). Entre sus prestaciones principales estas tarjetas soportan salidas digitales a relé y salidas analógicas en tensión y/o corriente (-10~10VDC, 4-20mA). La alimentación eléctrica será mediante el estándar industrial de 24 V aunque fueron diseñadas para funcionar en el rango de +10V a +30V DC.

El bus de control al igual que el de datos recorrerá perimetralmente el polígono de pruebas del GERA y se canalizará hasta una segunda pasarela RS485/Ethernet ubicada dentro del gabinete de red instalado en el segundo nivel del edificio de este centro. Para esto se

empleará cable de control trenzado y apantallado de dos hilos de cobre de 1.5 mm^2 cada uno. El mapa de direcciones de memoria de las tarjetas empleadas es listado en el Anexo # 6.

Bus de estaciones meteorológicas Thies-Clima:

Inicialmente se enlazarán tres estaciones meteorológicas de esta marca, aunque estará disponible la opción de incluir nuevas estaciones de diferentes fabricantes siempre y cuando el protocolo de comunicación empleado sea compatible con la norma física RS-485 y no incluya tramas que generen incompatibilidades con la trama de Thies-Clima. De las tres estaciones disponibles una se encuentra instalada en las instalaciones del GERA y dos son de tipo portátiles. La primera solo dispone de comunicación RS-232 por lo que será necesario emplear una pasarela adicional RS-232/RS-485 para poder canalizar la información hasta el bus de la pasarela RS-485/Ethernet que se encontrará ubicada dentro del gabinete de red del segundo piso del edificio del centro. Las dos estaciones restantes sí disponen de comunicación RS-485, pero con el objetivo de poder efectuar investigaciones fuera del centro se enlazarán al bus empleando un radio enlace con dos dispositivos ICP DAS SST-2450 (Ref. Epígrafe 1.1.2.4), lo que garantizará la funcionalidad de estas estaciones en un radio de 1 km fuera del centro.

2.1.2.2 Bloque de software

Software GERAOpc:

Función Principal, en esta se configurarán todos los parámetros de comunicación Ethernet y Serie. La comunicación sobre Ethernet será según el estándar normado por fundación OPC (OPC Foundation, 2014) y la comunicación serie será empleando el protocolo DCON. Se crearán 3 hilos paralelos para la ejecución del programa, uno para el intercambio de datos mediante el puerto serie (con nivel de prioridad máximo y un procesador dedicado), otro para la atención de solicitudes mediante la red Ethernet (nivel de prioridad medio) y un tercer hilo dedicado a atender los eventos generados por la interfaz gráfica.

Función actualizar canal, esta función será la encargada de acceder al bus DCON y entregar al servidor OPC la medición tomada. La frecuencia con que se ejecutará esta

función será variable y dependerá de un coeficiente de importancia de la medición calculado dinámicamente tal como se define en el Anexo # 7.

Función habilitar canal, mediante esta función el administrador del software podrá deshabilitar o habilitar los canales que de manera automática son detectados por el sistema. Esta opción posibilitará optimizar el acceso a los canales de medición cuando se requieran frecuencias de muestreo elevadas.

Función Comunicación Ethernet, dará tratamiento a los pedidos de medición efectuados vía TCP-IP o UDP mediante el estándar OPC.

Función chequear permisos, esta función se ejecutará cada vez que un usuario nuevo inicie sesión en el software. La aplicación cliente (GERALab) enviará de manera cifrada una lista con los permisos que se le deben conceder en el servidor (**GERAStation**)

Los Ítems listados por el servidor OPC de esta aplicación serán creados de manera automática en función de la cantidad de módulos de adquisición de datos instalados y podrán ser deshabilitados en cualquier momento por el administrador del sistema. Los comandos que conforman el protocolo de comunicación con los módulos serán leídos desde un fichero externo ".ini" de manera que modificar el protocolo de comunicación del software con los equipos remotos no implique cambios estructurales significativos. Se dispondrá además de un mecanismo que periódicamente efectúe un proceso de detección de nuevos dispositivos conectados al bus con el objetivo de dotar al mismo de características "Plug&Play"

Software GERAStation:

Función Principal, en esta se configurarán todos los parámetros de comunicación Ethernet y Serie. La comunicación sobre Ethernet será según el estándar normado por la fundación OPC y la comunicación serie será empleando el protocolo específico para la estaciones meteorológicas de la marca Thies-Clima. Esta función además establecerá un enlace con una Base de Datos MySQL para almacenar los datos obtenidos.

Función Comunicación Ethernet, dará tratamiento a los pedidos de medición efectuados vía TCP-IP o UDP mediante el estándar OPC, accederá al bus serie y entregará mediante la Ethernet el dato solicitado.

Función Tomar instantáneas, esta función se ejecutará cíclicamente cada 20 segundos y guardará en la base de datos todos los parámetros que entregan las estaciones meteorológicas enlazadas a la red con el objetivo de visualizar esta información temporalmente en el sitio web del GERA.

Función Tomar históricos, esta función se ejecutará cíclicamente cada 24 horas y descargará los valores promedios cada 10 minutos del día trascurrido que se encuentran almacenados en las memorias internas de las estaciones meteorológicas que conforman la red. Una vez efectuada esta operación se procederá a eliminar las mediciones temporales efectuadas por la función **Tomar instantáneas.**

Función chequear permisos, esta función posee exactamente las mismas características operativas que la homóloga para el software GERAOpc.

Software GERALab:

Función Principal, esta función se ejecutará al iniciar el software y establecerá el enlace con los dos servidores OPC (GERAOpc y GERAStation), abriendo un hilo de procesamiento para cada uno y un tercer hilo para dar tratamiento a los restantes eventos generados por el sistema. En esta función se incorporarán las comprobaciones de los permisos asignados al usuario mediante la lectura de un fichero llave, el cual será transmitido de manera íntegra a los servidores antes mencionados para que procedan a hacer efectivos los permisos en cuestión.

Función configurar experimento, esta función iniciará un asistente de configuración donde se le solicitará al usuario: seleccionar los canales de entradas y salidas asociados a su experimento, seleccionar la frecuencia con la que desea que se almacenen sus datos, seleccionar la ruta donde desee almacenar las mediciones del experimento y escribir un nombre para el experimento. En este punto es válido aclarar que el alcance deseado para el sistema no incluye opciones de cálculos o procesamientos estadísticos de la información adquirida. Debido a que la naturaleza de los datos adquiridos es muy diversa, este procesamiento final lo ejecutarán como estime conveniente el investigador o usuario.

Función Asociar canal, esta función ejecutará las configuraciones necesarias para que el usuario pueda modificar el origen de los datos mostrados por cualquier indicador en la pantalla del SCADA.

Función cerrar programa, esta función cerrará el programa solo si no se está ejecutando ningún proceso de medición en segundo plano.

Función mostrar valores, esta función se encargará de transferir los valores que se encuentran almacenados en variables a los componentes gráficos en la pantalla.

Función iniciar mediciones, una vez llamada esta función se comenzará la toma de mediciones y la ejecución de las acciones de control.

Función detener mediciones, una vez llamada esta función se detendrá la toma de mediciones y la ejecución de las acciones de control.

Función mover componentes, esta función habilitará la posibilidad de que el usuario pueda cambiar la posición de cualquier componente en el sinóptico.

Función cambiar dimensiones, esta función habilitará la posibilidad de que el usuario pueda cambiar el ancho y el alto de cualquier componente en el sinóptico.

Función eliminar componente, esta función habilitará la posibilidad de que el usuario pueda eliminar cualquier componente del sinóptico.

Función nuevo componente, esta función habilitará la posibilidad de que el usuario pueda agregar nuevos componentes gráficos al sinóptico.

Función guardar proyecto, esta función generará un fichero ".cfg" donde se almacenará la información tomada por la **Función configurar experimento** así como los sinópticos creados por el usuario.

Función abrir proyecto, esta función recibirá un fichero ".cfg" en el que estará almacenada todas las configuraciones y sinópticos del experimento en cuestión.

Función efectuar mediciones, esta función tomará los datos de los servidores OPC y los almacenará en las variables correspondientes del GERALab.

Función imagen de fondo, esta función permitirá asignar una imagen o esquema al fondo del mímico que se está modificando.

Función escribir en fichero .txt, esta función almacenará en la ruta previamente indicada las mediciones tomadas de los canales de entrada, los estados escritos en los canales de salidas y la hora exacta en que se tomó esta información.

Función configurar lazo de control, esta función iniciará un asistente de configuración donde el usuario podrá crear nuevos lazos de control asociando canales de entradas y canales de salidas mediante diferentes esquemas de control predefinidos además de configurar los principales parámetros de los controladores seleccionados. Una vez finalizada esta configuración se generará una nueva pantalla con el esquema en bloques del lazo creado, en la que se podrá modificar en cualquier momento los parámetros de ajustes del controlador.

Función manejo de alarmas, esta función permitirá al usuario visualizar, reconocer, eliminar y filtrar todos los eventos de alarmas generados por el sistema.

Función crear nueva alarma, esta función permitirá al usuario configurar las condiciones para que se genere un evento indicando alguna anormalidad en el experimento en desarrollo.

2.2 Formalización del sistema propuesto empleando Redes de Petri

Como se pudo constatar en el Capítulo anterior, actualmente existe una búsqueda de métodos formales para el modelado y diseño de sistemas de automatización en la comunidad científica, donde juega un papel importante la formalización, verificación y validación de los sistemas.

Siguiendo la metodología de diseño sobre GHENeSys propuesta por (Benitez, Silva, Villafurela, Gomis, & Sudriá, 2008), el primer aspecto a considerar en el ***paso1*** es el estudio del proceso a controlar y sus requerimientos funcionales, que se corresponde con la fase de **especificaciones informales.** Igualmente los ***pasos del 2 al 5*** de la metodología están estrechamente relacionados, y se asocian a la fase de **formalización** de las aplicaciones,

garantizando con ello la obtención del modelo idóneo. Estos pasos tienen como objetivo fragmentar el sistema en las menores unidades funcionales posibles, definir los niveles de jerarquía en la estructura del modelo e identificar sus interrelaciones entre los módulos (subredes) que la forman, correspondiéndose las unidades funcionales definidas con las subredes, tratando de representarlas lo más sencilla posible, según lo permita la aplicación dada. Siguiendo estos principios, cada aplicación que conforma el sistema fue descompuesta en las estructuras modulares mostradas en el Anexo # 8 y son descritas a continuación.

Software GERAOpc:

Compuesto por dos niveles jerárquicos y cuatro hilos concurrentes para la ejecución del programa. En el nivel superior se encuentra la unidad funcional "**Función Principal**" descrita en el apartado **2.1.2.2**; esta función será la encargada de crear los cuatro hilos de ejecución y administrar el acceso a las variables globales que serán utilizadas por los demás bloques funcionales. En el segundo nivel se encuentran las funciones encargadas de intercambiar información con el medio circundante, ya sea mediante el Bus de Datos, la Red Ethernet o la interfaz gráfica.

Como se puede apreciar este esquema fue concebido con la intención de lograr la mayor estructura modular posible de manera que se viabilice el modelado empleando el software V*isual Object Net 2.7,* aunque esto implique que la estructura concebida pueda cambiar. Vale además destacar que resulta difícil desde una etapa primaria del diseño prever la cantidad total de módulos, su nivel de jerarquía y sus interrelaciones, esto lo genera todo el estudio ulterior, depurándose y refinándose gradualmente a medida que el sistema toma mayor alcance y profundidad, para lo que se empleará la metodología de diseño Arriba-Abajo y Abajo-Arriba (Perl Design Patterns, 2008).

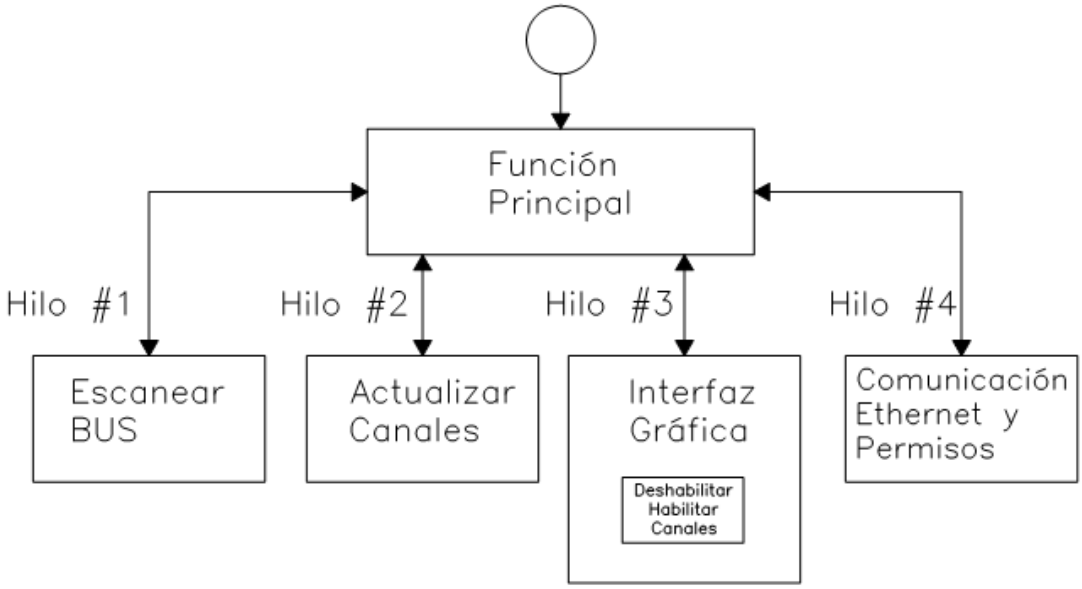


Figura 2.2: Estructura modular jerárquica del software GERAOpc

En la figura anterior y las posteriores se podrá observar que se ha representado el hilo de la interfaz gráfica pese a que esta función no ha sido relacionada en los puntos anteriores. Estas adiciones son debido a que el software Labwindows/CVI es un ambiente de desarrollo basado en ANSI C que de manera automática incluye esta función en todos los proyectos para dar solución al manejo de ventanas gráficas y eventos asociados desde la óptica de un lenguaje de programación no orientado a objetos.

Software GERAStation:

De manera similar al anterior, el software GERAStation estará compuesto por dos niveles jerárquicos y cuatro hilos concurrentes para la ejecución del programa. En el nivel superior se encuentra la unidad funcional "**Función Principal**" descrita en apartados anteriores; esta función será la encargada de crear los cuatro hilos de ejecución y administrar el acceso a las variables globales que serán utilizadas por los demás bloques funcionales. En el segundo nivel se encuentran las funciones encargadas de intercambiar información con el medio circundante, ya sea mediante el Bus de Datos, la Red Ethernet o la interfaz gráfica. Este esquema fue concebido con la intención de lograr la mayor estructura modular posible, lo que implica que la estructura concebida puede cambiar. Se destaca que el Hilo #1 y el # 2

disponen de un recurso de uso exclusivo (El bus de datos), por lo que se hace necesario algún mecanismo de bloqueo que no permita la entrada de ambas funciones en la misma unidad de tiempo.

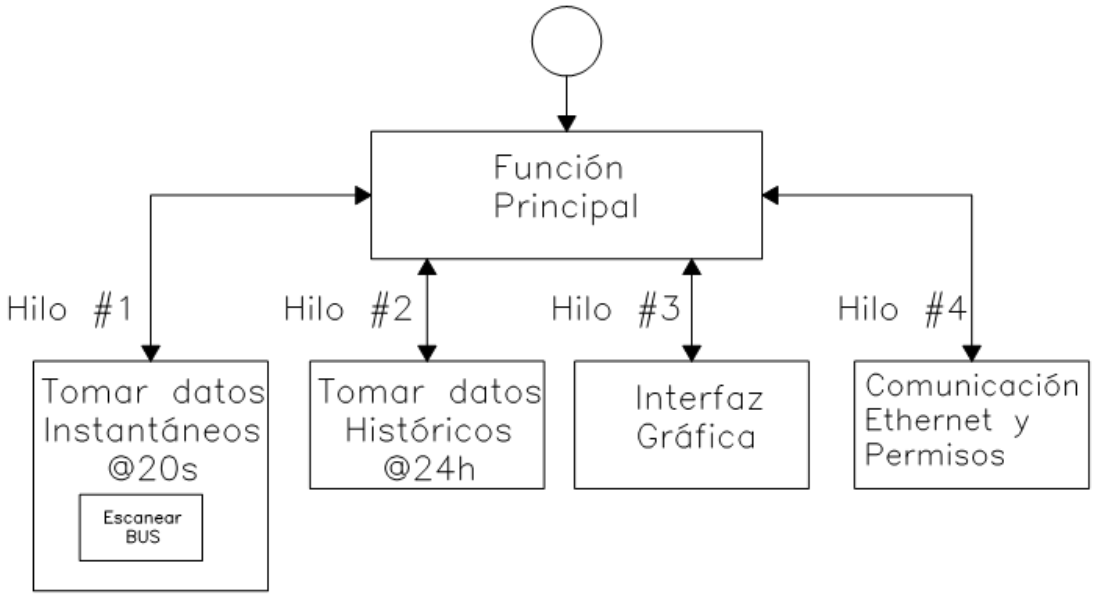


Figura 2.3: Estructura modular jerárquica del software GERAStation

Software GERALab:

El software GERALab estará compuesto por dos niveles jerárquicos y cinco hilos concurrentes para la ejecución del programa. En el nivel superior se encuentra la unidad funcional "**Función Principal**" descrita en apartados anteriores; esta función será la encargada de crear los hilos de ejecución y administrar el acceso a las variables globales que serán utilizadas por los demás bloques funcionales. En el segundo nivel se encuentran las funciones encargadas de intercambiar información con el medio circundante, ya sea enlazándose con los servidores OPC, respondiendo a los pedidos de la interfaz gráfica o generando información para ser almacenada en un fichero de texto y en una base de datos. Las funciones descritas en el apartado 2.1.2.2 que compondrán la aplicación GERALab formarán parte de los bloques funcionales mostrados en la Figura 2.4.

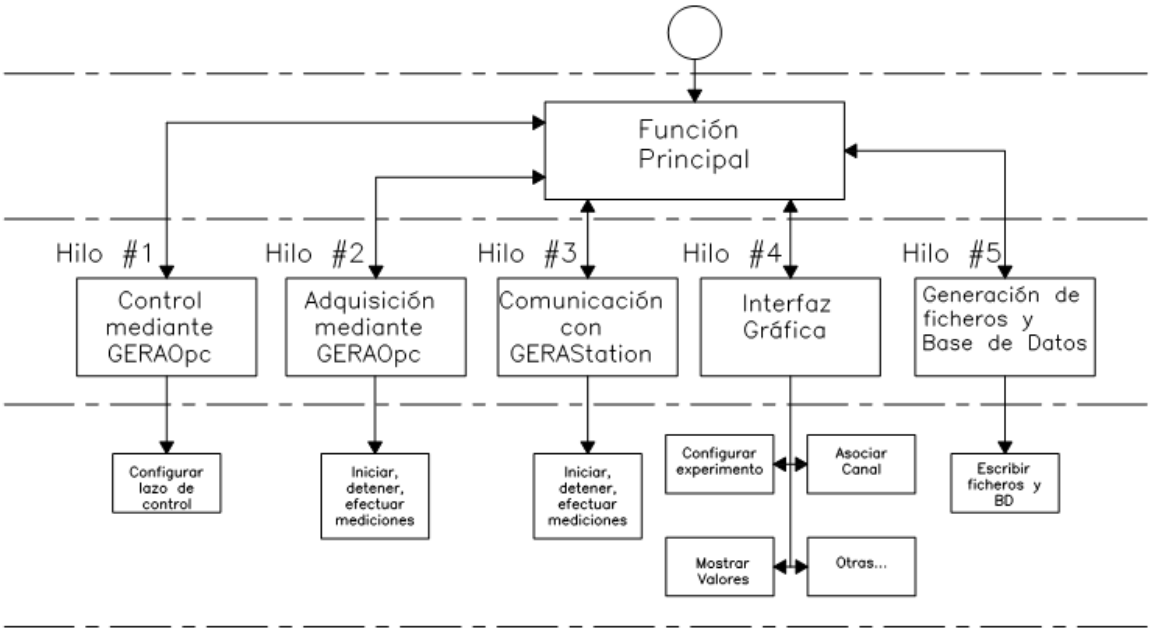


Figura 2.4: Estructura modular jerárquica del software GERALab con las principales funciones del sistema

2.2.1 Modelado en GHENeSys de la aplicación GERAOpc

Para realizar el modelado de la aplicación se empleó el diseño conceptual mostrado en la Figura 2.2, identificándose los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos

- Estado Inicial
- Función Principal
- Interfaz Gráfica
- Comunicación Ethernet
- Escaneo de BUS
- Actualizar E/S
- Estado Final

Eventos

- Ejecutar Aplicación
- Inicializar Función Principal
- Inicialización de 4 hilos de ejecución concurrentes
- Fin de ejecución del Hilo # 1
- Fin de ejecución del Hilo # 2
- Fin de ejecución del Hilo # 3
- Fin de ejecución del Hilo # 4
- Cerrar Aplicación

A partir del análisis del cumplimiento de los requerimientos funcionales, según la metodología empleada, se decidió modelar como subsistemas independientes todos los recursos identificados, excepto el Estado inicial y el Estado Final. No obstante en el esquema general (Figura 2.5) se emplearon modelos simplificados de los subsistemas previstos, donde solo aparecen los estados de inicio y fin de la ejecución de cada subsistema y por ende se modelarán las interacciones y empleo de recursos comunes entre subsistemas en puntos ulteriores.

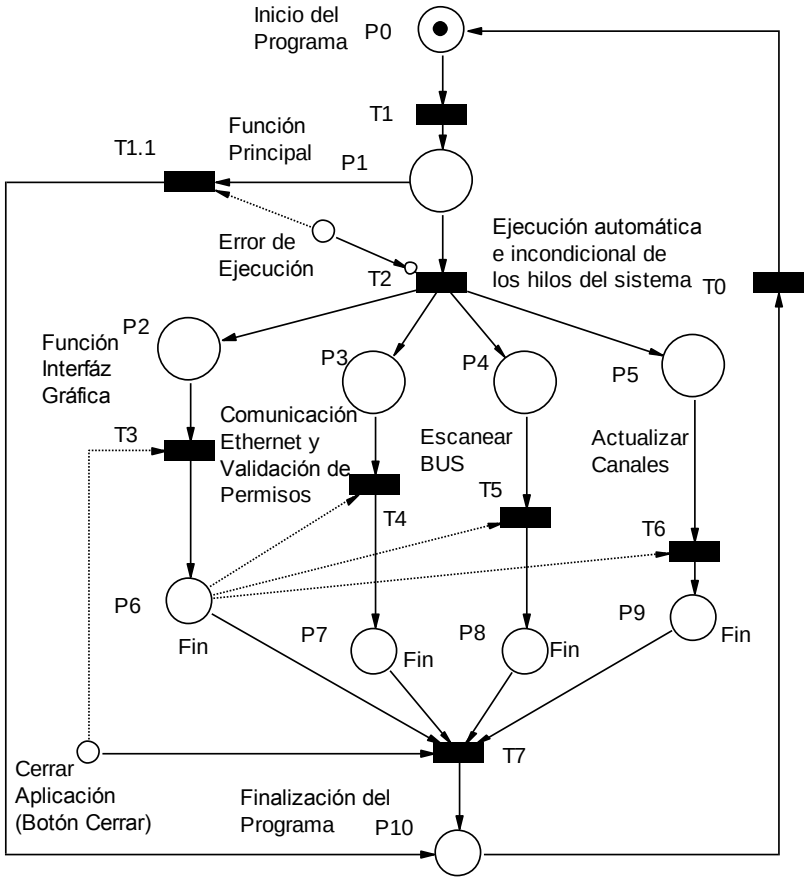


Figura 2.5: Modelo en PN del funcionamiento general de la aplicación GERAOpc

En este modelo el lugar **P0** se corresponde con el recurso “Estado Inicial” y es el punto donde comienza la ejecución del programa, a continuación y de manera incondicional se ejecuta la transición T1, dando paso al lugar **P1** es el que representa el recurso “Función principal”, los lugares **P2** y **P6** representan el recurso “Interfaz Gráfica”, los lugares **P3** y **P7**

representan el recurso “Comunicación Ethernet”, los lugares **P4** y **P8** representan el recurso “Escaneo de BUS”, los lugares **P5** y **P9** representan el recurso “Actualizar E/S” y el lugar **P10** representa el recurso “Estado Final” o cierre de la aplicación.

Como se puede apreciar mediante la transición **T2** se crean los 4 hilos de ejecución concurrentes previstos en la etapa conceptual del proyecto. Finalmente se emplea un lugar auxiliar para representar la acción del usuario que indica el cierre de la aplicación, donde una vez ejecutado dicho evento, se cierra la interfaz gráfica y se espera que los demás procesos finalicen sus labores para finalizar la aplicación.

Subsistema “Función Principal”:

Para realizar el modelado de la función principal se identificaron los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Carga de Comandos DCON	▪ Carga de Comandos Completada
▪ Conexión con la BD	▪ Conexión a BD Completada
▪ Configuración de Comunicaciones	▪ Config. de comunicación Completada
▪ Configuración de Hilos concurrentes	▪ Config. de Hilos Completada
▪ Ejecución de Hilos concurrentes	▪ Finalizando Aplicación

En el Anexo # 9 se muestra el modelo en PN obtenido a partir de los recursos y eventos listados para este subsistema y teniendo en cuenta los recursos analizados para el funcionamiento general del sistema. El modelo detallado exclusivo del lugar P1 es el encerrado dentro del recuadro mostrado.

Como se puede observar los lugares P0 y P10 del modelo general, han sido representados con el objetivo de ilustrar la sustitución del macro elemento P1 por la red detallada de la función principal. De manera que la función principal siempre se inicie en el lugar **P0** y

culmine con la ejecución de los 4 hilos concurrentes o con la finalización del sistema. Por lo que el modelo de este subsistema podrá ser reducido al empleado en el modelo general.

Subsistema "Interfaz Gráfica":

Para realizar el modelado de la función "Interfaz Gráfica" se identificaron los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Procesamiento del Evento Habilitar/Deshabilitar Canales	▪ Habilitar/Deshabilitar un canal

En el Anexo # 10 se muestra el modelo en PN obtenido a partir de los recursos y eventos listados para este subsistema y teniendo en cuenta los recursos analizados para el funcionamiento general del sistema. El modelo detallado exclusivo del lugar P2 es el encerrado dentro del recuadro mostrado.

Como se puede observar los lugares **P6** y **P10** del modelo general, han sido representados con el objetivo de ilustrar la sustitución del macro elemento P2 por la red detallada de la función interfaz gráfica. De manera que la función interfaz gráfica siempre se inicie en **T2** y culmine con la señal de cierre de la aplicación en **T3**. Por lo que el modelo de este subsistema podrá ser reducido al empleado en el modelo general.

Esta red inicia su ejecución en **T2** y se mantendrá cíclicamente atendiendo los eventos generados por el usuario (Habilitar o Deshabilitar Canales) hasta que se ejecute el comando cerrar aplicación y por ende se habilite la transición **T3** indicando que se cierre la interfaz gráfica y se deje de atender los restantes eventos.

Subsistema "Comunicación Ethernet":

Para realizar el modelado de la función "Comunicación Ethernet" se identificaron los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Validación de Permisos ▪ Denegación de Información ▪ Enlace con la BD ▪ Respuesta a Pedidos ▪ Tareas Completadas	▪ Solicitud de un usuario nuevo ▪ Solicitud de un usuario registrado ▪ Permisos aceptados ▪ Permisos denegados ▪ Respuesta a permisos denegados ▪ Conexión con BD establecida ▪ Respuesta a solicitud de medición ▪ Inicio de nuevo ciclo

En el Anexo # 11 se muestra el modelo en PN obtenido a partir de los recursos y eventos listados para este subsistema y teniendo en cuenta los recursos analizados para el funcionamiento general del sistema. El modelo detallado exclusivo del lugar P3 es el encerrado dentro del recuadro mostrado.

Como se puede observar los lugares **P7** y **P10** del modelo general, han sido representados con el objetivo de ilustrar la sustitución del macro elemento P3 por la red detallada de la función Comunicación Ethernet. De manera que la función Comunicación Ethernet siempre se inicie en **T2** y culmine con el cierre de la aplicación en **T4**. Por lo que el modelo de este subsistema podrá ser reducido al empleado en el modelo general.

Esta red inicia su ejecución en **T3** y esperará hasta que se produzca la solicitud de mediciones de un usuario (funcionamiento por el método de interrupciones), en caso de no ser un usuario registrado se procederá a validar los permisos asignados, en caso contrario se responderá a la solicitud con la medición correspondiente. El cierre de la función (o estado **P7**) se producirá cuando se habilite la transición **T4**.

En este modelo se representaron como lugares auxiliares P6, P8, P9 y Cerrar Aplicación, para modelar la interacción de este subsistema con los restantes, por lo que para validar el correcto funcionamiento de la red obtenida se deberán obviar estos lugares.

Subsistemas "Escaneo de BUS" y "Actualizar E/S":

Para el modelado de las funciones relacionadas con el acceso directo a los buses de datos y control se identificaron los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Uso de los buses de comunicación ▪ Envío y recepción de Información mediante los buses ▪ Almacenamiento en BD ▪ Liberación de los buses	▪ Inicio de función escanear buses ▪ Fin de función escanear buses ▪ Inicio de función actualizar canales ▪ Fin de función actualizar canales ▪ Buses ocupados ▪ Almacenando en BD ▪ Inicio de nuevo ciclo

En el Anexo # 12 se muestra el modelo en PN obtenido a partir de los recursos y eventos listados para estos subsistemas y teniendo en cuenta los recursos analizados para el funcionamiento general del sistema. Los modelos detallados exclusivos de los lugares P4 y P5 se encuentran demarcados con líneas discontinuas.

Como se puede observar los lugares **P6, P8, P9** y **P10** del modelo general, han sido representados con el objetivo de ilustrar la sustitución de los macro elementos P4 y P5 por las redes detalladas de las funciones Escanear BUS y Actualizar Canales. De manera que las funciones se inicien en **T2** y culminen respectivamente y culminen con el cierre de la aplicación mediante **T5** y **T6**. Por lo que los modelos de estos subsistemas podrán ser reducidos al empleado en el modelo general.

Esta red inicia su ejecución en **T2** y se ejecutará cíclicamente hasta que se habilite la transición T7 (comando de cierre enviado por el usuario). Se puede observar el empleo del recurso compartido **P.5-9.6** que modela el acceso a los buses y garantiza que primero se ejecute la función "Escaneo de BUS" para luego dar paso a la "Actualización de canales".

En este modelo se representaron como lugar auxiliar P6 y Cerrar Aplicación, para modelar la interacción de estos subsistemas con los restantes, por lo que para validar el correcto funcionamiento de la red obtenida se deberán obviar estos lugares.

2.2.2 Modelado en GHENeSys de la aplicación GERAStation

Para realizar el modelado de la aplicación se empleó el diseño conceptual mostrado en la Figura 2.3, identificándose los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos

- Estado Inicial
- Función Principal
- Interfaz Gráfica
- Comunicación Ethernet
- Lectura de Parámetros Instantáneos
- Lectura de Parámetros históricos
- Estado Final

Eventos

- Ejecutar Aplicación
- Inicializar Función Principal
- Inicialización hilos de ejecución concurrentes
- Fin de ejecución del Hilos
- Cerrar Aplicación

A partir del análisis del cumplimiento de los requerimientos funcionales, según la metodología empleada, no es necesaria la verificación de propiedades de los subsistemas independientes, puesto que se consideran iguales a los homónimos del sistema GERAOpc donde se asume que son estructural y dinámicamente iguales. Por lo que las funciones que se implementarán en ANSI C para el software GERAOpc serán reutilizadas íntegramente en el Software GERAStation, haciendo uso de una de las características más potentes de este lenguaje de programación que es la reutilización de código.

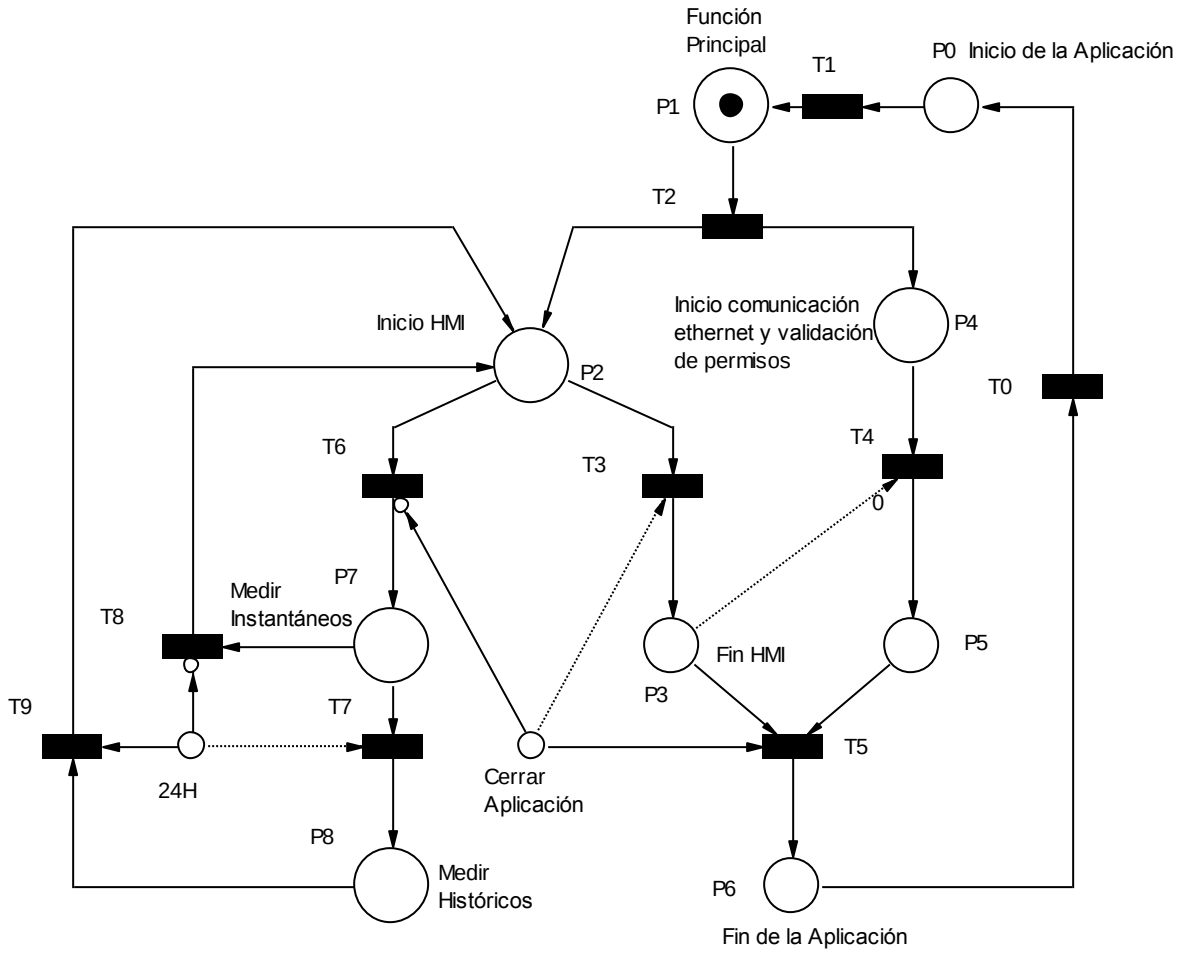


Figura 2.6: Modelo en PN del funcionamiento general de la aplicación GERAStation

En este modelo el lugar **P0** se corresponde con el recurso “Estado Inicial” y es el punto donde comienza la ejecución del programa. El lugar **P1** es el que representa el recurso “Función principal”, los lugares **P2, P3, P7** y **P8** representan el recurso “Interfaz Gráfica”, dentro del cual se ejecutarán las tareas de medición, los lugares **P4** y **P5** que representan el recurso “Comunicación Ethernet” y el lugar **P6** que representa el recurso “Estado Final” o cierre de la aplicación.

Como se puede apreciar mediante la transición **T2** se crean los hilos de ejecución concurrentes previstos en la etapa conceptual del proyecto. Finalmente se emplea un lugar auxiliar para representar la acción del usuario que indica el cierre de la aplicación, donde una vez ejecutado dicho evento, se cierra la interfaz gráfica y se espera que los demás procesos finalicen sus labores para finalizar la aplicación. Y un segundo lugar auxiliar para representar el transcurso de 24 horas y disparar la acción de almacenar registros históricos.

2.2.3 Modelado en GHENeSys de la aplicación GERALab

Para realizar el modelado de la aplicación se empleó el diseño conceptual mostrado en la Figura 2.5, identificándose los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Estado Inicial ▪ Función Principal ▪ Interfaz Gráfica ▪ Sistema de Control ▪ Sistema de adquisición # 1 ▪ Sistema de adquisición # 2 ▪ Estado Final	▪ Ejecutar Aplicación ▪ Inicializar Función Principal ▪ Inicialización de 4 hilos de ejecución concurrentes ▪ Fin de ejecución del Hilo # 1 ▪ Fin de ejecución del Hilo # 2 ▪ Fin de ejecución del Hilo # 3 ▪ Fin de ejecución del Hilo # 4 ▪ Cerrar Aplicación

A partir del análisis del cumplimiento de los requerimientos funcionales, según la metodología empleada, se decidió modelar como subsistemas independientes los recursos “Sistema de Control”, “Sistema de adquisición # 1”. En el esquema general (Figura 2.7) se emplearon modelos simplificados de los subsistemas previstos, donde solo aparecen los estados de inicio y fin de la ejecución de cada subsistema y por ende se modelarán las interacciones y empleo de recursos comunes entre subsistemas en puntos ulteriores.

En este modelo el lugar **P0** se corresponde con el recurso “Estado Inicial” y es el punto donde comienza la ejecución del programa. El lugar **P1** es el que representa el recurso “Función principal”, los lugares **P2, P3** y **P4** representan el recurso “Sistema de Control”, los lugares **P5, P6** y **P6** representan el recurso “Sistema de adquisición # 1”, los lugares **P8, P9** y **P10** representan el recurso “Sistema de adquisición # 2”, los lugares **P11, P12** y **P13** representan el recurso “Interfaz Gráfica” y el lugar **P14** representa el recurso “Estado Final” o cierre de la aplicación.

Como se puede apreciar mediante las transiciones **T2 y T3** se crean los 4 hilos de ejecución concurrentes según corresponda a los permisos asignados al usuario. Finalmente se emplea un lugar auxiliar para representar la acción del usuario que indica el cierre de la aplicación, donde una vez ejecutado dicho evento, se cierra la interfaz gráfica y se espera que los demás procesos finalicen sus labores para finalizar la aplicación.

Como puntualización final cabe destacar que el modelo fue diseñado para que los cuatro hilos concurrentes se inicien independientemente de los permisos otorgados al usuario. En el caso del hilo asignado a la función de control, este se mantendrá en espera del cierre de la aplicación o de cambios en los permisos si al iniciar la aplicación GERALab se detectó que el usuario solo podría efectuar acciones de medición.

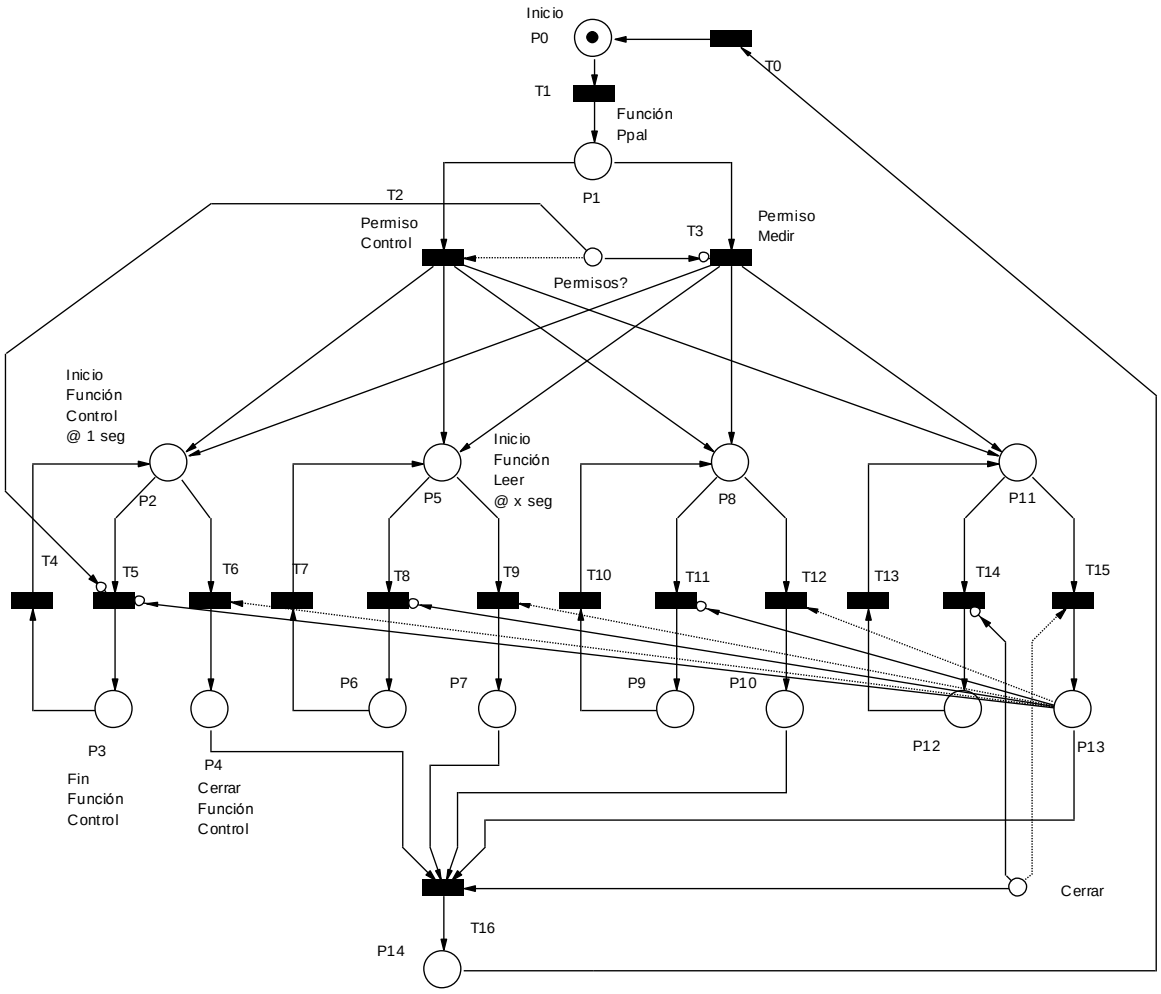


Figura 2.7: Modelo en PN del funcionamiento general de la aplicación GERALab

Subsistema “Adquisición # 1”:

Para realizar el modelado de la función “Adquisición # 1” se identificaron los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Tomar datos del servidor OPC ▪ Guardar en Base de Datos ▪ Guardar fichero de texto	▪ Inicio de la toma de datos ▪ Toma de datos correcta ▪ Toma de datos error ▪ Almacenamiento en BD correcto ▪ Almacenamiento en BD error

En el Anexo # 13 se muestra el modelo en PN obtenido a partir de los recursos y eventos listados para este subsistema y teniendo en cuenta los recursos analizados para el funcionamiento general del sistema.

Como se puede observar los lugares **P5**, **P6** y **P7** fueron representados para ilustrar la sustitución de los macro elementos por las redes detalladas. Por lo que el modelo de este subsistema podrá ser reducido al empleado en el modelo general.

Subsistema "Control":

Para realizar el modelado de la función "control" se identificaron los siguientes recursos y eventos controlables que se generan a partir de estos y que intervienen en su funcionamiento.

Recursos	Eventos
▪ Planificación de tareas de control ▪ Calculo de controladores ▪ Envío de acciones de control	▪ Inicio de la toma de datos ▪ Toma de datos correcta ▪ Toma de datos error ▪ Almacenamiento en BD correcto ▪ Almacenamiento en BD error

En el Anexo # 14 se muestra el modelo en PN obtenido a partir de los recursos y eventos listados para este subsistema y teniendo en cuenta los recursos analizados para el funcionamiento general del sistema.

Como se puede observar los lugares **P2**, **P3** y **P4** fueron representados para ilustrar la sustitución de los macro elementos por las redes detalladas. Por lo que el modelo de este subsistema podrá ser reducido al empleado en el modelo general.

2.3 Verificación del sistema modelado

En este epígrafe además de efectuar la verificación de los modelos desarrollados empleando los métodos de reducción e invariantes, se realizó la validación de los modelos por simulación al comprobar el comportamiento funcional de cada módulo en todas sus variantes (funcionamiento normal, ante fallos, etc.) demostrándose el cumplimiento de los requerimientos funcionales mediante su interpretación física. Se comprueba la evolución funcional de los estados de las redes de acuerdo a la habilitación o deshabilitación de las transiciones controlables según los valores de los lugares auxiliares (pseudoboxes) que representan los estados de los buses, selectores, botones o marcas internas del proceso.

Para todos los casos se perfeccionó el funcionamiento de los modelos permitiendo la última versión de los mismos que se presenta en este trabajo. Esto se realiza según los pasos 9 y 10 de la metodología (Benitez, Silva, Villafurela, Gomis, & Sudriá, 2008) que indica la simulación para comprobar todo el funcionamiento de los requerimientos funcionales y luego la reclasificación y verificación de propiedades de cada modelo.

2.3.1 Verificación por el método de técnicas de reducción o descomposición

Aplicación GERAOpc:

Tomando como referencia la Figura 2.5, desechando el lugar auxiliar y los arcos habilitadores, se obtiene la PN principal correspondiente al modelo no controlado de la Aplicación GERAOpc. En este caso podemos observar que los pares de lugares P2-P6, P3-P7, P4-P8 y P5-P9 pueden fusionarse en uno solo aplicando la regla de fusión de lugares en serie (**FSP),** quedando la red como se muestra en el Anexo # 15 (a). Luego la red resultante

puede ser reducida nuevamente a la mostrada en el Anexo # 15 (b) una vez que se fusionan los cuatro lugares que se encuentran en paralelo (**FPP**) y finalmente aplicando sucesivamente las reglas FSP, FST y FSP se obtiene el resultado final de la reducción Anexo # 15 (c) donde claramente se observa que el software transita del estado inicial P0 al estado final P10 mediante la transición T0. Por lo que de una manera simple se observa que la red resultante es viva, limitada y segura (Murata, 1989), por lo que la red del funcionamiento general del sistema GERAOpc también lo es.

Subsistema "Función Principal":

Tomando como referencia el modelo mostrado en el Anexo # 9 y desechando el lugar auxiliar utilizado para modelar la ejecución correcta o no de las funciones, se obtiene la PN correspondiente al modelo no controlado de la función Principal de la Aplicación GERAOpc.

En el caso del modelo general no se tuvo en cuenta la posibilidad de que el subsistema "Función Principal" pudiera finalizar la ejecución de la aplicación por lo que para proceder a la reducción de la red y el chequeo de propiedades se asume que la transición **T.1-10.1** es independiente de este modelo.

Finalmente y aplicando las reglas de reducción analizadas en la Figura 1.4 se obtiene el modelo simplificado que se muestra en el Anexo # 9 (b) donde se puede constatar la coincidencia con el modelo general empleado y de una manera simple se observa que la red resultante es viva, limitada y segura (Murata, 1989), por lo que la red del Subsistema "Función Principal" también lo es.

Subsistema "Interfaz Gráfica":

Tomando como referencia el modelo mostrado en el Anexo # 10 y desechando el lugar auxiliar utilizado para modelar la señal de cierre de la aplicación, se obtiene la PN correspondiente al modelo no controlado de la función "Interfaz Gráfica" de la

Aplicación GERAOpc. Seguidamente se aplicaron las reglas de reducción correspondientes y se obtuvo el modelo simplificado que se muestra en el Anexo # 10 (b) donde se puede constatar la coincidencia con el modelo general empleado y de una manera simple se observa que la red resultante es viva, limitada y segura (Murata, 1989), por lo que la red del Subsistema "Interfaz Gráfica" también lo es.

Subsistema "Comunicación Ethernet":

Tomando como referencia el modelo mostrado en el Anexo # 11 y desechando los seis lugares auxiliares empleados se obtiene la PN correspondiente al modelo no controlado de la función "Comunicación Ethernet" de la Aplicación GERAOpc. Luego mediante la aplicación de las reglas de reducción correspondientes se obtuvo el modelo simplificado que se muestra en el Anexo # 11 (b) donde se puede constatar la coincidencia con el modelo general empleado y de una manera simple se observa que la red resultante es viva, limitada y segura (Murata, 1989), por lo que la red del Subsistema "Comunicación Ethernet" también lo es.

Subsistemas "Escaneo de BUS" y "Actualizar E/S":

Teniendo en cuenta que la red diseñada es la que mayor complejidad presenta en este sistema se determinó solo efectuar la verificación aplicando el enfoque de la matriz de incidencia, ya que no se considera viable el empleo de técnicas de reducción o descomposición y de manera análoga se obtendrán los mismos resultados.

Aplicación GERAStation:

Tomando como referencia la Figura 2.6, desechando los lugares auxiliares y los arcos habilitadores e inhibidores, se obtiene la PN principal correspondiente al modelo no controlado de la Aplicación GERAStation. En este caso se aplicaron las reglas FPT con las transiciones T6 y T8, FST con las transiciones T7 y T9, FST con las transiciones resultantes

T6T8 y T7T9, EST con la transición resultantes T6T8T7T9 y el lugar P2, FSP con el lugar resultante P2T6T8T7T9 y el lugar inicial P3 y FSP con los lugares P4 y P5 para obtener el modelo reducido mostrado en el Anexo # 16. Como se puede observar si se aplicara finalmente las FPP y FSP se evidenciaría que el software transita del estado inicial P0 al estado final P6 mediante la transición T0. Por lo que de una manera simple se observa que la red resultante es viva, limitada y segura (Murata, 1989), por lo que la red del funcionamiento general del sistema GERAStation también lo es.

Subsistema "Comunicación Ethernet":

Como se puntualizó anteriormente, en el caso de este subsistema se empleará la misma implementación para el subsistema equivalente de la aplicación GERAOpc.

Aplicación GERALab

La aplicación GERALab por su complejidad e importancia es la aplicación central del sistema SCADA a desarrollar, y por ende la red que mayor complejidad presenta en su modelo, por tanto se determinó solo efectuar la verificación aplicando el enfoque de la matriz de incidencia, ya que no se considera viable el empleo de técnicas de reducción o descomposición y de manera análoga se obtendrán los mismos resultados.

2.3.2 Verificación aplicando el enfoque de la matriz de incidencia

En este caso, aunque el sistema es pequeño, lo que permite aplicar sólo las técnicas de reducción, se aplicará la verificación mediante la matriz de incidencia para demostrar la efectividad del método y el cumplimiento de algunos requerimientos funcionales del sistema.

Aplicación GERAOpc:

Con la ayuda de la herramienta de modelado PIPE, se realizó la verificación por el enfoque de matriz de incidencia de la red de funcionamiento general de la aplicación GERAOpc, obteniéndose las matrices e invariantes mostradas más adelante. En el caso de los subsistemas que componen el sistema GERAOpc se determinó solo aplicar esta técnica al modelo de Subsistemas "Escaneo de BUS" y "Actualizar E/S" (Anexo #12), no siendo necesaria la verificación aplicando el enfoque de la matriz de incidencia a los demás modelos, debido a que anteriormente se demostró que estos son reducibles al modelo general empleado que se verificará a continuación:

Matriz de incidencia combinada:

	T1	T2	T3	T4	T5	T6	T7	T0
P0	-1	0	0	0	0	0	0	1
P1	1	-1	0	0	0	0	0	0
P2	0	1	-1	0	0	0	0	0
P3	0	1	0	-1	0	0	0	0
P4	0	1	0	0	-1	0	0	0
P5	0	1	0	0	0	-1	0	0
P6	0	0	1	0	0	0	-1	0
P7	0	0	0	1	0	0	-1	0
P8	0	0	0	0	1	0	-1	0
P9	0	0	0	0	0	1	-1	0
P10	0	0	0	0	0	0	1	-1

Matriz de invariantes de transición (T):

T1	T2	T3	T4	T5	T6	T7	T0
1	1	1	1	1	1	1	1

Donde se observa que la red está cubierta por una invariante positiva de transición, por lo que se puede considerar limitada y viva. Esto demuestra que todas las transiciones de la red pueden ser habilitadas en algún momento de la dinámica del sistema (ausencia de bloqueos) y que todas las transiciones se habilitan sólo una vez en cada ciclo del funcionamiento del sistema.

La interpretación física es que todos los elementos de la red están, al menos, en un estado activo (vivos) sin bloquearse ni mantenerse en lazos indefinidos porque siempre se habilita la transición siguiente. Por tanto se garantiza la operación de forma eficiente y segura como se solicita en los requerimientos funcionales del sistema, dotando así a la aplicación de la eficiencia y seguridad requerida para la ejecución de las características de flexibilidad y concurrencia analizadas en el capítulo anterior.

Matriz de invariantes de lugar (P):

P0	P1	P10	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	1	1	1	0	0	0	1	0	0	0	1
1	1	1	0	1	0	0	0	1	0	0	1
1	1	1	0	0	1	0	0	0	1	0	1
1	1	1	0	0	0	1	0	0	0	1	1

En esta matriz se observa que la red está cubierta por invariantes positivas de lugar, por lo que se considera limitada o definida. Esto demuestra que el proceso general de la aplicación GERAOpc se ejecutará secuencialmente y que al menos uno de los estados siempre estará habilitado con un marcado máximo de un "*token*", lo cual se puede representar mediante las siguientes ecuaciones que se corresponden con los cuatro hilos que se ejecutarán concurrentemente:

$$M\ P0 + M\ P1 + M\ P2 + M\ P6 + M\ P10 = 1$$

$$M\ P0 + M\ P1 + M\ P3 + M\ P7 + M\ P10 = 1$$

$$M\ P0 + M\ P1 + M\ P4 + M\ P8 + M\ P10 = 1$$

$$M\ P0 + M\ P1 + M\ P5 + M\ P9 + M\ P10 = 1$$

La interpretación física de esta invariante es que el sistema siempre estará en uno de sus cuatro hilos o parado, lo cual corresponde al cumplimiento del requerimiento funcional previsto.

Aplicación GERAStation:

Con la ayuda de la herramienta de modelado PIPE, se realizó la verificación por el enfoque de matriz de incidencia de la red de funcionamiento general de la aplicación GERAStation, obteniéndose las matrices e invariantes mostradas a continuación.

Matriz de incidencia combinada:

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
P1	0	1	-1	0	0	0	0	0	0	0
P2	0	0	1	-1	0	0	-1	0	1	1
P3	0	0	0	1	0	-1	0	0	0	0
P4	0	0	1	0	-1	0	0	0	0	0
P5	0	0	0	0	1	-1	0	0	0	0
P6	-1	0	0	0	0	1	0	0	0	0
P7	0	0	0	0	0	0	1	-1	-1	0
P8	0	0	0	0	0	0	0	1	0	-1
P0	1	-1	0	0	0	0	0	0	0	0

Matriz de invariantes de transición (T):

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	1	0	1	0

Donde se observa que la red está cubierta por tres invariantes positivas de transición, por lo que se puede considerar limitada y viva. Esto demuestra que todas las transiciones de la red pueden ser habilitadas en algún momento de la dinámica del sistema (ausencia de bloqueos) y que todas las transiciones se habilitan sólo una vez en cada ciclo del funcionamiento del sistema.

La interpretación física es que todos los elementos de la red están, al menos, en un estado activo (vivos) sin bloquearse ni mantenerse en lazos indefinidos porque siempre se habilita la transición siguiente. Por tanto se garantiza la operación de forma eficiente y segura como se

solicita en los requerimientos funcionales del sistema, permitiendo también en este caso que se cumplan con eficiencia y calidad las características de flexibilidad y concurrencia analizadas en el capítulo anterior.

Matriz de invariantes de lugar (P):

P1	P2	P3	P4	P5	P6	P7	P8	P0
1	1	1	0	0	1	1	1	1
1	0	0	1	1	1	0	0	1

En esta matriz se observa que la red está cubierta por invariantes positivas de lugar, por lo que se considera limitada o definida. Esto demuestra que el proceso general de la aplicación GERAStation se ejecutará secuencialmente y que al menos uno de los estados siempre estará habilitado con un marcado máximo de un "*token*", lo cual se puede representar mediante las siguientes ecuaciones que se corresponden con los dos hilos reales que finalmente se ejecutarán concurrentemente:

$$\mathrm{M\ P0} + \mathrm{M\ P1} + \mathrm{M\ P2} + \mathrm{M\ P3} + \mathrm{M\ P6} + \mathrm{M\ P7} + \mathrm{M\ P8} = 1$$

$$\mathrm{M\ P0} + \mathrm{M\ P1} + \mathrm{M\ P4} + \mathrm{M\ P5} + \mathrm{M\ P6} = 1$$

La interpretación física de esta invariante es que el sistema siempre estará en uno de los dos hilos o parado, lo cual corresponde al cumplimiento del requerimiento funcional previsto ya que dentro del hilo interfaz gráfica se dará tratamiento a las funciones "tomar instantáneas" y "tomar datos históricos".

Aplicación GERALab:

Con la ayuda de la herramienta de modelado PIPE, se realizó la verificación por el enfoque de matriz de incidencia de la red de funcionamiento general de la aplicación GERALab, obteniéndose las matrices e invariantes mostradas más adelante. En el caso de los subsistemas que lo componen se determinó innecesario el empleo de esta técnica debido a

que anteriormente se demostró que estos son reducibles al modelo general empleado que se verificará a continuación. Partiendo de la Matriz de incidencia combinada para el modelo de GERALab mostrada a continuación se obtienen las siguientes matrices de invariantes:

Matriz de Incidencia GERALab:

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
P0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P1	0	1	-1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	0	0	1	1	1	-1	-1	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	-1	1	0	0	0	0	0	0	0	0	0	0	0
P4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-1
P5	0	0	1	1	0	0	0	1	-1	-1	0	0	0	0	0	0	0
P6	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	0	0
P7	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	-1
P8	0	0	1	1	0	0	0	0	0	0	1	-1	-1	0	0	0	0
P9	0	0	0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0
P10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1
P11	0	0	1	1	0	0	0	0	0	0	0	0	0	1	-1	-1	0
P12	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	0
P13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1
P14	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Matriz de invariantes de transición (T):

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	0	1	1
1	1	0	1	0	0	1	0	0	1	0	0	1	0	0	1	1
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Donde se observa que la red está cubierta por seis invariantes positivas de transición, por lo que se puede considerar limitada y viva. Esto demuestra que todas las transiciones de la red pueden ser habilitadas en algún momento de la dinámica del sistema (ausencia de bloqueos)

y que todas las transiciones se habilitan sólo una vez en cada ciclo del funcionamiento del sistema o subsistemas.

La interpretación física es que todos los elementos de la red están, al menos, en un estado activo (vivos) sin bloquearse ni mantenerse en lazos indefinidos porque siempre se habilita la transición siguiente. Por tanto, también para la aplicación principal se garantiza la operación de forma eficiente y segura de las características de flexibilidad y concurrencia analizadas en el capítulo anterior como se solicita en los requerimientos funcionales del sistema.

Matriz de invariantes de lugar (P):

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
1	1	0	0	0	0	0	0	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	0	0	0	0	0	0	1
1	1	0	0	0	1	1	1	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	1	1	1	0	0	0	1

En esta matriz se observa que la red está cubierta por invariantes positivas de lugar, por lo que se considera limitada o definida. Esto demuestra que el proceso general de la aplicación GERALab se ejecutará secuencialmente y que al menos uno de los estados siempre estará habilitado con un marcado máximo de un "*token*", lo cual se puede representar mediante las siguientes ecuaciones que se corresponden con los cuatro hilos reales que finalmente se ejecutarán concurrentemente:

$$\mathrm{M\ P0} + \mathrm{M\ P1} + \mathrm{M\ P11} + \mathrm{M\ P12} + \mathrm{M\ P13} + \mathrm{M\ P14} = 1$$

$$\mathrm{M\ P0} + \mathrm{M\ P1} + \mathrm{M\ P2} + \mathrm{M\ P3} + \mathrm{M\ P4} + \mathrm{M\ P14} = 1$$

$$\mathrm{M\ P0} + \mathrm{M\ P1} + \mathrm{M\ P5} + \mathrm{M\ P6} + \mathrm{M\ P7} + \mathrm{M\ P14} = 1$$

$$\mathrm{M\ P0} + \mathrm{M\ P1} + \mathrm{M\ P8} + \mathrm{M\ P9} + \mathrm{M\ P10} + \mathrm{M\ P14} = 1$$

La interpretación física de estas invariantes es que el sistema siempre estará en uno de los cuatro hilos o parado, lo cual corresponde al cumplimiento del requerimiento funcional

previsto ya que dentro del hilo interfaz gráfica se dará tratamiento a todos los eventos generados por el usuario.

2.4 Implementación en LabWindows/CVI del sistema

Para el adecuado desarrollo de las aplicaciones que componen el sistema, la verificación y validación formal de los modelos en GHENeSys ayudaron grandemente a la reducción de errores de diseño y la implementación eficiente, así como el empleo de todos los recursos de programación concurrente disponibles en el ambiente de desarrollo LabWindows/CVI.

2.4.1 Aplicación GERAOpc

La implementación de este software fue desarrollada bajo el ambiente LabWindows/CVI 10.0. El proyecto está compuesto por tres ficheros, GERAOpc.uir donde se almacena toda la información referente al diseño de la interfaz gráfica y los ficheros GERAOpc.h y GERApc.c donde se declararon y se implementaron todas las funciones respectivamente.

Mediante el modelo en GHENeSys (Figura 2.5) se logró evaluar los pasos de ejecución de la secuencia de funcionamiento y las condicionales que generaban estos cambios de estado, lo cual fue utilizado para esta implementación de todas las funciones en los ficheros GERAOpc.h y GERAOpc.c. Por ejemplo, la sincronización de la ejecución de los cuatro hilos concurrentes y el predomino de la “Interfaz Gráfica”, sobre los recursos “Comunicación Ethernet”, “Escaneo de BUS, y “Actualizar E/S”, para luego ejecutar el cierre de la aplicación. Además los detalles internos representados en cada subred analizada en este modelo principal de GERAOpc fueron tenidos en cuenta para la programación de todas estas funciones. Ej. Función de Acceso al Bus de Datos.

La verificación de propiedades y validación funcional de estos modelos garantizo que se transmitieran estas características al programa garantizando eficiencia y seguridad de la ejecución concurrente de estas funciones. Ej. Ejecución de múltiples hilos intercomunicados.

La interfaz diseñada incluye un aspecto minimalista con el objetivo de facilitar las características de flexibilidad, pero se incluyen potentes algoritmos de auto-detección de canales y visualización de los datos obtenidos, facilitando al usuario solo dos áreas de interacción y un área de información (Anexo # 17). En el área de interacción # 1 (marcada en rojo) se mostrarán todos los módulos de adquisición y control detectados en los buses de datos y adjunto a cada canal el usuario dispondrá de un "*Check Box*" y una etiqueta, mediante el primero podrá habilitar o deshabilitar el empleo del canal en cuestión y mediante la segunda podrá visualizar en tiempo real los valores instantáneos de cada canal.

Mediante el área de interacción # 2 (marcada en verde) se podrá introducir una demora entra cada ciclo de medición o barridos de canales, esto con el objetivo de no sobrecargar en exceso los recursos disponibles en el servidor que soporte este software. Y el área de visualización (marcada en azul) estará compuesta por dos simples indicadores lumínicos, uno que parpadeará en color rojo para indicar cuando se está accediendo a los buses de datos y el otro indicador parpadeará en azul para indicar cuando se están ejecutando las rutinas de almacenamiento en la base de datos de tiempo real.

Dentro de las funciones que dispone LabWindows para el trabajo con hilos concurrentes, manejo de datos en tiempo real y acceso a bases de datos, se comentan a continuación algunas de las empleadas en el desarrollo del GERAOpc.

Función CmtNewThreadPool:

Esta función recibe como parámetros de entrada dos valores de tipo entero y devuelve un valor que servirá para detectar la correcta ejecución de la función. El primer valor de entrada será el máximo número de hilos concurrentes que se podrán ejecutar en cada bloque concurrente, para determinar este valor en función de las prestaciones reales del hardware disponible se emplea la siguiente fórmula obtenida de (National Instruments Corporation, 2010), que en nuestro caso arrojó un valor de 4:

$$\#\ Maximo\ Hilos = 2 + (2 * \#\ de\ procesadores)$$

Este cálculo corresponde plenamente con el análisis realizado en el modelo GERAOpc (Fig. 2.5) de los cuatro hilos concurrentes.

El segundo valor de entrada es un puntero des-referenciado a una variable de tipo entero, lo que le permitirá a la función escribir en esta dirección de memoria, cuyo valor entregado se utilizará para hacer referencia al hilo de ejecución creado.

Se emplean además las funciones **CmtScheduleThreadPoolFunction** y **CmtReleaseThreadPoolFunctionID** para inicializar y finalizar respectivamente el hilo de ejecución configurado con **CmtNewThreadPool.**

Función DBConnect:

Esta función recibe como parámetro una cadena de texto con el nombre del origen de datos o DSN en que se almacenará la información (Microsoft Corporation, 2013), manteniendo así un alto nivel de portabilidad. Facilitando el caso que se necesite cambiar el tipo de base de datos o la ubicación de estas, solo será necesario cambiar la cadena de texto que recibe **DBConnect.** Dotando así a la aplicación de características de flexibilidad adicionales.

Asociadas a esta función se emplean además las funciones **DBImmediateSQL** y **DBDisconnect** que facilitan el uso de comandos SQL para el manejo de la base de datos (ISO, 2008) y la desconexión de esta respectivamente.

2.4.2 Aplicación GERAStation

Al igual que GERAOpc la implementación de GERAStation fue desarrollada bajo el ambiente Labwindows/CVI 10.0. El proyecto está compuesto por tres ficheros, Station.uir donde se almacena toda la información referente al diseño de la interfaz gráfica y los ficheros Station.h y Station.c donde se declararon y se implementaron todas las funciones respectivamente.

De igual forma que en GERAOpc, el modelo de la Figura 2.6 permitió una implementación eficiente y segura de las funciones, su concurrencia y sincronización. Ej. Sincronización de mediciones instantáneas y toma de datos históricos**.**

A través de la verificación de propiedades y validación funcional del modelo se garantizó que en la implementación de esta función se garantizara la estabilidad y el correcto funcionamiento empleando recursos compartidos.

La interfaz diseñada incluye los aspectos necesarios para garantizar la ergonomía y flexibilidad requerida por el usuario. En la implementación se incluyeron algoritmos de auto-detección de estaciones Thies-Clima, facilitando al usuario un área de interacción gráfica y un área de visualización (Anexo # 18). En el área de interacción (marcada en rojo) se mostrarán todas las estaciones meteorológicas enlazadas a la red. Mediante el área de visualización (marcada en verde) el administrador del programa podrá chequear de manera rápida si los valores obtenidos son lógicos.

Dentro de las funciones que dispone Labwindows para el trabajo con hilos concurrentes, manejo de datos en tiempo real y acceso a bases de datos, se comentan a continuación algunas de las empleadas en el desarrollo del GERAStation que por su importancia son vitales en el funcionamiento del mismo.

CreateUDPChannelConfig:

Esta función recibe seis parámetros de entrada, entre ellos se debe especificar el puerto UDP a través del cual se escucharán las solicitudes de medición y un puntero a la función que dará tratamiento a la trama de datos empleando el estándar OPC (OPC Foundation, 2014). Asociada a esta función, también se empleó la función **CreateUDPChannel** que permite inicializar el canal tomando las configuraciones declaradas con **CreateUDPChannelConfig** y la librería de funciones "**DataSocket Library**" para hacer efectiva la transferencia de información mediante el mecanismo "*Object Linking and Embedding*" OLE adaptado al control de procesos (OPC Foundation, 2014)

OpenComConfig:

Mediante el empleo de esta función se abre el puerto serie para establecer la comunicación con las estaciones, ya que las pasarelas empleadas son del tipo TCP/IP-RS485 el puerto serie al que se hace referencia es un puerto serie virtual que enruta los datos enviados por el software **GERAStation** al puerto UDP de la pasarela a la escucha (FabulaTech Corporation, 2014).

2.4.3 Aplicación GERALab

La implementación de este software fue desarrollada bajo el ambiente Labwindows/CVI 10.0. El proyecto está compuesto por múltiples ficheros, entre los que destacan GERALab.uir donde se almacena toda la información referente al diseño de la interfaz gráfica, asynctmr.h que es la librería que incluye las funciones de temporizadores asincrónicos, inifile.h que posibilita la lectura automatizada de fichero .ini y movectrl.h que facilita la creación y edición de componentes en tiempo de ejecución. Este desarrollo es considerado la versión 2.0 del GERALab ya que tuvo su precedente comentado en el capítulo anterior. Por limitaciones objetivas y de alcance del sistema el número máximo de componentes que el usuario puede agregar a la pantalla es 200 y el número máximo de canales de E/S es 500, aunque con pequeñas modificaciones estructurales el soporte se puede ampliar teóricamente hasta infinito.

Por medio del modelo de la Fig. 2.7 se logró estudiar cuales eran las condicionales e interacciones entre las diferentes funciones de este programa, como por ejemplo la sincronización de la toma de datos de diferentes servidores OPC. A través de la verificación de propiedades y validación funcional del modelo se garantizó que esta implementación funcionara adecuadamente y se lograra obtener la solución técnica al problema planteado.

La interfaz diseñada incluye un aspecto sencillo con el objetivo de facilitar las características de flexibilidad, pero se incluyen complejas rutinas de comunicación con los servidores OPC, visualización de los datos obtenidos, almacenamiento de datos y cálculos de controladores, facilitando al usuario solo un área de interacción inicial ubicada en la parte superior derecha

de la ventana de la aplicación (Anexo # 19 (A)). Mediante estas opciones el usuario podrá iniciar un asistente donde le indicará al sistema todos los detalles del experimento en cuestión, tales como nombre, tiempo de muestreo, canales de E/S a utilizar, etc. Seguidamente el usuario podrá configurar el mímico del experimento, con la posibilidad de insertar cinco tipos diferentes de componentes de visualización en pantalla hasta un límite de 200 indicadores en una misma pantalla. Para cada uno de los componentes insertados el usuario tendrá disponibles las opciones de cambiar tamaño, posición, etiqueta, origen de datos o eliminarlo (Anexo # 19 (B)).

La otra opción que se encuentra disponible solo para usuarios con permisos especiales es la creación de lazos de control. Inicialmente se encuentran habilitados los principales esquemas de control a lazo cerrado y una opción para generar condiciones de control. Para la ejecución de estas tareas se dispuso de un "*Timer*" asincrónico que se ejecuta cíclicamente con un tiempo base de 1 segundo, y en cada ciclo se determina mediante un sistema de "*prescalers*" a que lazo de control les corresponde actualizar sus salidas.

Las condiciones de control están conformadas bajo la siguiente regla:

- *SI* **[CANAL SELECCIONADO#1]** *ES* **[MAYOR|MENOR|IGUAL]** *QUE* **[VALOR FIJO|CANAL SELECCIONADO#2]** ENTONCES **[CANAL SELECCIONADO#X] = [0|1| VALOR FIJO|CANAL SELECCIONADO#X+= VALOR FIJO]**

Los controladores disponibles son:

- ON-OFF + Histéresis + Zona Muerta
- PID

Los esquemas de control disponibles son:

- Lazo cerrado común
- Lazo cerrado en casada

El tercer grupo de opciones está destinado a la configuración de alarmas, las cuales podrán ser configuradas mediante la siguiente regla:

- *SI* **[CANAL SELECCIONADO#1]** *ES* **[MAYOR|MENOR|IGUAL]** *QUE* **[VALOR FIJO|CANAL SELECCIONADO#2]** *ENTONCES ACTIVAR ALARMA* **[TEXTO]**

El cuarto grupo de opciones da la posibilidad de guardar y cargar los experimentos en cuestión y en el quinto y último grupo se encuentran las opciones de iniciar o detener el experimento.

Dentro de las funciones disponibles en LabWindows para el trabajo con hilos concurrentes, manejo de datos en tiempo real y acceso a bases de datos, se comentan a continuación algunas de las empleadas en el desarrollo del GERALab.

MakeMovableCtrl:

Hace que un componente en pantalla pueda ser movido y/o cambiado de tamaño en tiempo de ejecución. Esta función es clave para las características de flexibilidad propuestas como objetivo de este trabajo, ya que le facilita al usuario final del software posibilidades que en sistemas SCADAS clásicos no se disponen, facilitando la creación o modificación de HMI's sin necesidad de ser usuarios desarrolladores del sistema.

NewAsyncTimer:

Crea en un nuevo hilo un temporizador asincrónico que se ejecuta con alta prioridad. Esta función permite ejecutar en tiempo real operaciones de control que independientemente de la cola de prioridades que establece el sistema operativo se priorizan las acciones programadas en este temporizador.

NewAsyncTimerWithPriority:

Esta función opera de manera similar a la anterior, pero con la peculiaridad que facilita fijar el nivel de importancia o prioridad con que se ejecutará la porción de código indicada.

2.5 Validación del sistema mediante la explotación y uso profesional

Producto a que ningún método de verificación y validación es totalmente perfecto se pasó entonces a la etapa de validación de la aplicación real del sistema.

Se comprobó el comportamiento funcional de cada módulo en todas sus variantes (funcionamiento normal, ante fallos, etc.) demostrándose el cumplimiento de los requerimientos funcionales mediante su interpretación física.

Se comprueba la evolución funcional de los estados de la red de acuerdo a la habilitación o deshabilitación de las transiciones controlables según los valores de los lugares auxiliares (pseudoboxes) que representan los estados de los sensores, selectores, botones o marcas internas del proceso y del conteo de los tiempos en las transiciones temporizadas.

Para todos los casos se perfeccionó el funcionamiento de los modelos permitiendo la última versión de los mismos que se presenta en este trabajo. Esto se realiza según los pasos 9 y 10 de la metodología (Benitez, Silva, Villafurela, Gomis, & Sudriá, 2008) que indica la simulación para comprobar todo el funcionamiento de los requerimientos funcionales y luego la reclasificación y verificación de propiedades de cada modelo.

Un ejemplo de lo antes mencionado es la ejecución sincronizada de las funciones de control y toma de datos en el software GERALab para obtener un fichero o base de datos que permite a los investigadores analizar el comportamiento del objeto estudiado tomando como base la misma unidad de tiempo para representar los parámetros medidos y las acciones de control ejecutadas, siendo esta la principal potencialidad que ofrece este sistema para centros de investigación o laboratorios de adquisición de datos.

Finalmente la nueva versión del sistema SCADA Flexible ha sido empleada por más de dos años en el GERA como la herramienta central para el desarrollo de experimentos y toma de datos en investigaciones y proyectos de carácter nacional e internacional, arrojando resultados exitosos que son avalados según se muestra en el documento adjunto en el Anexo # 20.

Conclusiones Parciales

- La aplicación de las redes PN como método formal permitió obtener un modelo a partir del cual se genera el código de las tres aplicaciones de software que componen el sistema.
- Mediante el análisis de propiedades se logró comprobar en el proceso de diseño la validez de los modelos formales obtenidos en cuanto a vivacidad y limitación.
- Con la simulación de los modelos obtenidos se corroboraron varias de las propiedades verificadas, permitiendo además validar los requerimientos funcionales del sistema en todos sus estados de operación.
- Se diseñó e implementó el sistema SCADA y se comprobó su funcionamiento mediante el uso profesional durante más de dos años.

CONCLUSIONES

1. Se analizaron las principales características de los sistemas SCADAS Flexibles para laboratorios y se concluyó que en el proceso de desarrollo es imprescindible el empleo de herramientas de modelado y verificación antes de su implementación.
2. Se determinó que Las Redes de Petri Jerárquicas Extendidas GHENeSys es la herramienta de diseño formal que mejor se adecua al proceso de desarrollo de SCADAs de experimentación en centros de investigación. Obteniendo así modelos validados a partir de los cuales se generó el código de las tres aplicaciones de software que componen el sistema.
3. Se seleccionó el ambiente de desarrollo LabWindows/CVI para la implementación del sistema, tomando como base las facilidades del mismo para cumplir con las funcionalidades requeridas para la creación de sistemas SCADA Flexibles.
4. Se implementó el sistema SCADA Flexible tomando como base los modelos desarrollados en GHENeSys.
5. Se comprobó su funcionamiento mediante el uso profesional durante más de dos años.

RECOMENDACIONES

1. Garantizar compatibilidad nativa del sistema con una variedad mayor de dispositivos de adquisición control existentes en el mercado.
2. Abordar el tema de la selección de elementos sensores de manera que en las mediciones obtenidas pueda ser analizado el error que introducen estos dispositivos y cuando sea procedente atenuarlos o eliminarlos de manera automática mediante algoritmos de software.
3. Adicionar al sistema los mecanismos para la visualización ¨off-line¨ de tendencias así como un mecanismo para la obtención de reportes y análisis estadísticos, de manera que no sea necesario el empleo de herramientas externas para este propósito.
4. Bajo la misma filosofía de sistema SCADA Flexible se recomienda habilitar al sistema GERALab para su funcionamiento bajo buses de adquisición de datos mejor ajustados al procesamiento de grandes volúmenes de información a altas frecuencias, tales como Ethernet, PCI Express o PXI Express. .
5. Transferir las tareas y algoritmos de control a los dispositivos remotos distribuidos en el campo, ya que en este sistema se efectúan estos procedimientos en las computadoras de los usuarios del sistema.

BIBLIOGRAFÍA

Baier, C., & Kaoten, J. (2008). *Principles of Model Checking.* Massachusetts, USA: The MIT Press Cambridge.

Bailey, D., & Wright, E. (2003). *Practical SCADA for Industry.* Pert: Elsevier-Newnes.

Barry, A. (2005). *Concurrent Programming.*

Benítez, I., Gomis, O., & Sudriá, A. (2008). Flexible manufacturing cell SCADA system for educational purposes. *Journal Computer Applications in Engineering Education, 16*(1), 50-62.

Benitez, I., Silva, J., Villafurela, L., Gomis, O., & Sudriá, A. (2008). Modeling extended Petri nets compatible with GHENeSys IEC61131 for industrial automation. *The international Journal of Advanced Manufacturing Technology, 36*, 1180-1190.

Bonet, P., Llado, C., Puijaner, R., & Knottenbelt, W. (2007). PIPE v2.5: A Petri Net Tool for Performance Modelling. *Proc. 23rd Latin American Conference on Informatics*, 50-56.

Clarke, E., & Wing, J. (1996). *Formal Methods: State of the Art and Future Directions.* Carnegie Mellon University.

D'Armas, A. (2012). *Sistema SCADA para el Laboratorio del GERA.* Santiago de Cuba: Universidad de Oriente.

David, R., & Alla, H. (1992). *Petri Nets and Grafcet – Tools for modelling Discrete Event Systems.* New York: Prentice hall.

Desel, J., & Esparza, J. (1995). *Free Choice Petri Nets.* London: Cambridge University Press.

Dingle, N., Knottenbelt, W., & Suto, T. (2009). PIPE2: A Tool for the Performance Evaluation of Generalised Stochastic Petri Nets. *ACM SIGMETRICS Performance Evaluation Review, 36*(4), 34-36.

Drath, R. (1997). *A Mathematical Approach to Describing a Class of Hybrid Systems.* Genf.: IEEE Workshop on Parallel and distributed Real Time Systems.

Drath, R. (1998). *Hybrid Object Nets: An Object Oriented Concept for Modeling Complex Hybrid Systems.* ADPM'98: Hybrid Dynamical Systems. 3rd International Conference on Automation of Mixed Processes.

Du, Y., Jiang, C., & Zhou, M. (2009). A Petri Net-Based Model for Verification of Obligations and Accountability in Cooperative Systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 299-308.

FabulaTech Corporation. (2014). *Serial Port Redirector Help.* FabulaTech Corporation.

Flemming, N. (2005). *ML with concurrency: design analysis implementation, and application.*

Formal Systems. (23 de Noviembre de 2001). *Formal Systems Web Site*. Obtenido de http://www.formal.demon.co.uk

Frey, G. (2000). Analysis of Petri Net based Control Algorithms - Basic Properties. *Proceedings of the American Control Conference ACC 2000*, 19-23.

Frey, G., & Litz, L. (2000). *Formal Methods in PLC Programming.* Nashville: Proceedings of the IEEE.

Girault, C., & Valk, R. (2001). *Petri Nets for Systems Engineering. A guide to Modelling, Verification, Validation, and applications.* Berlin: Springer-Verlag.

Glez, P., & Silva, R. (2001). *GHENSYS: Uma rede estendida para a modelagem, analise e projeto de sistemas complexos.* Sao Paulo: Proc. of SBAI2001.

Holloway, L., & Krogh, B. G. (1997). A Survey of Petri Net Methods for Controlled Discrete Event Systems. *Journal of Discrete Event Dynamic Systems*, 1850-1862.

ICP DAS Co., Ltd. (2009). I-7000 DIO Manual Rev:B1.3. ICP DAS.

ISO. (2008). *ISO/IEC 9075: Information technology, Database languages, SQL, Part 1: Framework (SQL/Framework).* New York: ISO/IEC.

Jiménez, E. (Enero de 2000). Redes de Petri de Supervisión y Simulación de Procesos Industriales Automatizados. *XXI Jornadas de Automática*. CEA-IFAC.

Jiménez, E. (2006). Monitorización, implementación y simulación de procesos industriales con SCADAS . *Universidad de La Rioja, Área de Ingeniería de Sistemas y Automática*, 50-62.

Lee, D., & DiCesare, F. (1994). Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Trans on Robotics and Automation*, 123-132.

Medling, J. (2009). Empirical Studies in Process Model Verification. *K. Jensen and W. van der Aalst* , 208-224.

Microsoft Corporation. (2013). *Microsoft Soporte*. Recuperado el 20 de Enero de 2013, de http://support.microsoft.com/kb/966849/es

Murata, T. (1989). Petri Nets: Properties, analysis and applications. *Proceedings of IEEE, 77*(4).

National Instruments. (2011). *How to Choose the Right DAQ Hardware for Your Measurement System.* Texas: National Instruments.

National Instruments Corporation. (2010). *LabWindows™/CVI™ Help.* Texas: National Instruments.

Olivera, A. (2009). *MSc. Thesis: Herramienta para la enseñanza de modelado de sistemas de automatización con redes de Petri GHENeSys.* Santiago de Cuba: Universidad de Oriente.

OPC Foundation. (Septiembre de 2014). *The Interoperability Standard for Industrial Automation*. Obtenido de http://opcfoundation.org

Parallel Systems Engineering. (20 de Noviembre de 2011). *Parallel systems engineering Web Site*. Obtenido de http://wotug.ukc.ac.uk

Perl Design Patterns. (2008). *Perl Design Patterns free book.* PriorityHealth.

Rodríguez, A. (2007). *Sistemas SCADA* (Segunda ed.). Barcelona: Marcombo S.A.

Sanz, M. (2013). *Las Redes de Petri como herramienta formal en el diseño de sistemas de automatización integrada.* Tesis en opción al Título de Master en Ciencias, Universidad de Oriente, Santiago de Cuba.

Schiper, A. (1989). *Concurrent Programming.* London: North Oxford Academic.

Silva, J., Miyagi, P., Matos, L., & Afsarmanesh, H. (1995). *A High Level Net to the Modeling of Discrete Manufacturing Systems.* London: IFIP/Hapman & Hall.

Simonak, S., Hudak, S., & S., K. (2009). Protocol Specification and Verification Using Process Algebra and Petri Nets. *International Conference on Computational Intelligence, Modelling and Simulation*, 110-114.

Telecommunications Industry Association. (1 de Marzo de 2005). *TIA Advancing Global Communications*. Obtenido de http://ftp.tiaonline.org/tr-30/TR-30.2/Public/2005%20Meetings/2005-03-Tampa/20503002-Draft%202%20Revised%20TSB-89.doc

Thies Clima. (9 de Mayo de 2014). *Thies Clima, the world of weather data*. Obtenido de http://www.thiesclima.com/Datalogger_DLx-MET_e.html

Tommila, T., Juhani, H., & Lauri, J. (2005). *Next generation of industrial automation: Concepts and architecture of a component-based control system.* Espoo: VTT Tiedotteita.

Uzam, M. (1998). *Ph. D. Thesis. Petri Net Based Supervisory Control of Discrete Event Systems and their Ladder Logic Diagram Implementations.* Salford: University of Salford.

Welling, L., & Thomson, L. (2005). *Desarrollo Web con PHP y MySQL.* Anaya Multimedia.

Xing, K., Hu, B., & Chen, H. (2006). Deadlock avoidance policy for Petri-net modelling of flexible manufacturing systems with shared resources. *IEEE Trans on Automatic Control*, 289-294.

Zapata, G. (2007). *Diseño de automatismos secuenciales para controladores lógicos programables.* Medellín: Universidad Nacional de Colombia.

ANEXO # 1: Principales dispositivos ICP y ADAM

No.	**Código**	**Tipo de Entradas**
1	ADAM-3112	Voltaje AC
2	ADAM-3114	Corriente AC
3	ADAM-4017	V, mV, mA
4	ADAM-4018	Termopares J, K, T, E, R, S, B
5	ICP-7017	V, mV, mA
6	ICP-7033	Termo resistencias
7	ICP-7041	Entradas Digitales
8	ICP-7034	Hasta 5 Amperes

ANEXO # 2: Protocolo DCON

Formato de los comandos: **(encabezado)(dirección)(comando)(CHK)(cr)**

Formato de las respuestas: **(encabezado)(dirección)(datos)(CHK)(cr)**

Dónde:

[**CHK**]: Dos caracteres de Suma de verificación

[**cr**]: Caracter de fin de comando o código ASCII de Retorno (0x0D)

Set de Comandos principales		
Comando	**Respuesta**	**Descripción**
%AANNTTCCFF	!AA	Fijar configuración del módulo
#**	Sin Respuesta	Muestreo de sincronización
#AA	>(Dato)	Leer Entrada Digital
#AAN	>(Dato)	Leer Entrada Digital del canal N
#AABBDD	>	Salida Digital
#AAN	!AA(Dato)	Leer entrada digital de un contador
$AA0	!AA	Ejecutar calibración de span
$AA1	!AA	Ejecutar calibración de cero
$AA2	!AATTCCFF	Leer configuración
$AA4	!S(Dato)	Leer datos sincronizados
$AA5	!AAS	Leer estado del reset
$AA6	!(Dato)	Leer estado de los canales
$AAF	!AA(Dato)	Leer versión del Firmware
$AAM	!AA(Dato)	Leer nombre del módulo
$AAC	!AA	Borrar entrada digital del “Latch”
$AACN	!AA	Borrar contador de entrada digital
$AALS	!(Dato)	Leer entrada digital del “Latch”
@AA	>(Dato)	Leer Entrada digital
@AA(Dato)	>	Escribir salida digital
~AAO(Dato)	!AA	Fijar nombre del módulo

ANEXO # 3: Protocolo Modbus

Los módulos de control empleados están provistos de una interfaz serie RS-485 y se comunican con mediante el protocolo MODBUS, donde el dispositivo master inicia la comunicación la trama requerida para efectuar operaciones de lectura o escritura. Cada trama está compuesta por 4 bloques.

- Dirección del dispositivo esclavo: es un número entre 1 y 255 que identifica a cada dispositivo en la red.
- Código de la función (comando): define la acción a ejecutar (Ej. Leer n palabras, escribir una palabra)
- Datos a enviar: define los parámetros de la función (Ej. Dirección de la palabra a escribir, valor de la palabra escribir, etc.)
- Palabra de control (CRC): es usado para detectar errores en la trasmisión.

Funciones Modbus:

- Leer n palabras (función 03)
- Escribir una palabra (Función 06)

Formato de las tramas:

Dirección	Función	Dirección		Número de palabras		CRC	
1 byte	1 byte	2 bytes		2 bytes		2 bytes	
Desde 1 hasta 255	03	MSB	LSB	MSB	LSB	MSB	LSB

Anexo # 4: Ficha Técnica estación Thies Clima **DLx MET**

Valores estándar de entradas:

- Contador de 16 bit para todos los anemómetros de tipo THIES
- Serie sincrónico para todos los sensores de dirección del viento de tipo THIES
- 2 entradas para Pt100
- 0~1V para sensores de humedad relativa
- 0~5V para sensores de presión atmosférica
- 0~40mV para sensores de radiación solar con constante de calibración
- 0/4~20mA para sensores estándar
- Interfaz de comunicación RS-232/RS-485/USB

Conversor A/D:

- Resolución de 24 bit con entradas diferenciales

Ciclo de medición:

- Seleccionable entre 1 segundo ~ 60 minutos

Alimentación:

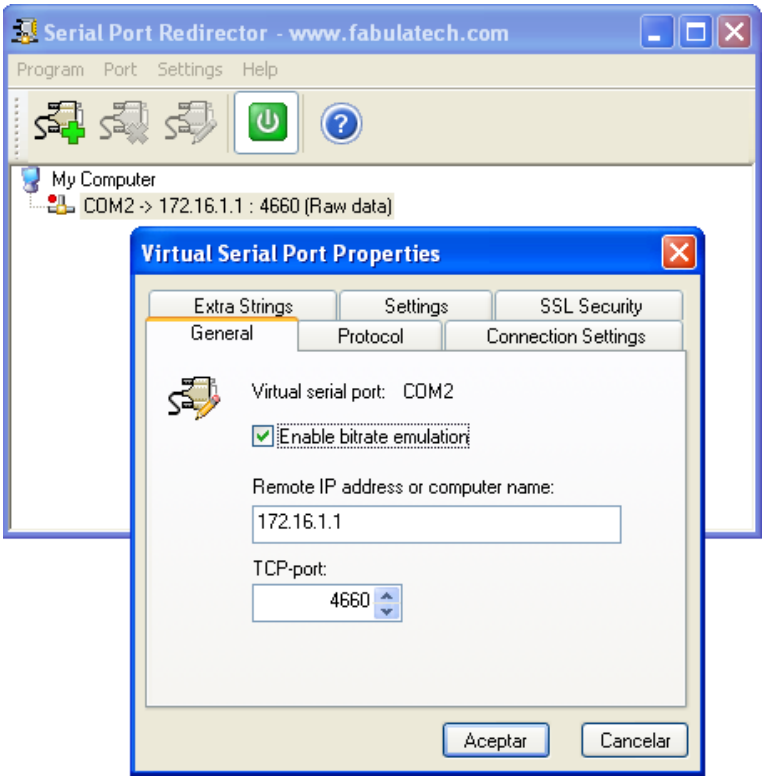
- Batería 12 V / 7 Ah
- Panel Fotovoltaico 12V / 20 W
- Alterna 115 / 230 V

Temperatura de operación:

- -30 ~ 60 ^{o}C

Anexo # 5: Puerto Serie Virtual

Debido a que estos dispositivos se enlazan con la PC mediante comunicación Ethernet, se hace necesario el empleo de puertos Series Virtuales que faciliten la interfaz entre estos equipos y el software SCADA. De manera que con el empleo del software “Serial Port Redirector” es posible enrutar la información enviada a un puerto serie determinado hasta una dirección IP.



ANEXO # 6: Mapa de memorias tarjetas propietarias GERA

Dirección	Bytes	Variable
0280h	**2**	Salida -10~10 VDC # 1
0282h	**2**	Salida -10~10 VDC # 2
0284h	**2**	Salida -10~10 VDC # 3
0286h	**2**	Salida 4-20 mA # 1
0288h	**2**	Salida 4-20 mA # 2
028Ah	**2**	Salida 4-20 mA # 3
0292h	**2**	Salida 0-100% Variador de Velocidad 0.4kW
0293h	**1**	Salida Relé # 1
0294h	**1**	Salida Relé # 2
0295h	**1**	Salida Relé # 3
0296h	**1**	Salida Relé # 4
0297h	**1**	Salida Relé # 5

ANEXO # 7: Cálculo del coeficiente de importancia de los canales de medición

El coeficiente de importancia de cada canal de medición será denotado por **K** y será empleado de manera directa y proporcional para determinar la posición en la cola de ejecución de la toma de datos. Este valor será recalculado antes de ejecutar cada nuevo ciclo de mediciones.

$$K = \frac{Ua * Tr}{Tm * Nc}$$

Dónde:

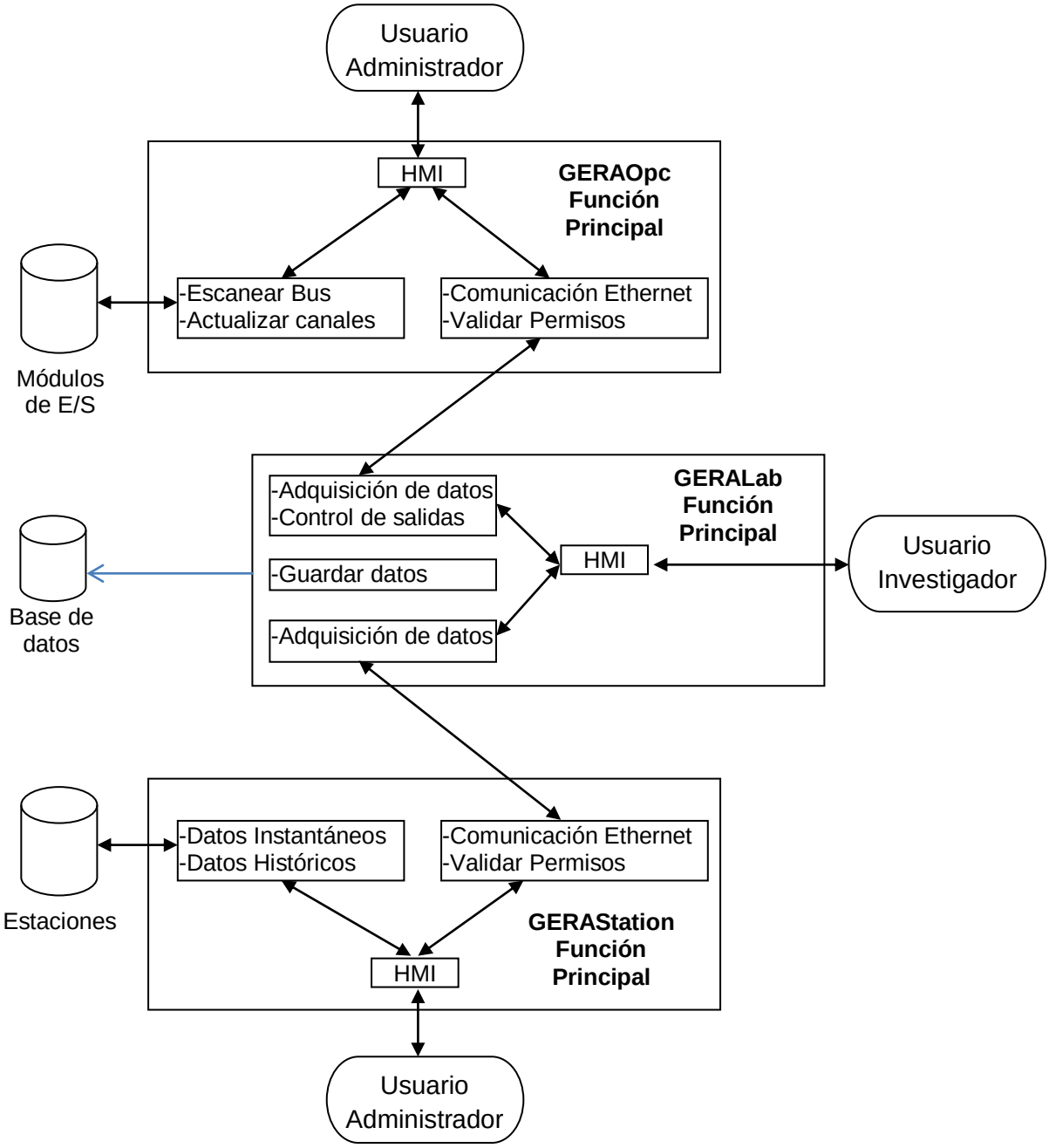
Ua: Cantidad de usuarios asociados al canal

Tr: Tiempo de respuesta del canal, dado en *mS*

Tm: Tiempo medio entre solicitudes al canal, dado en *S*

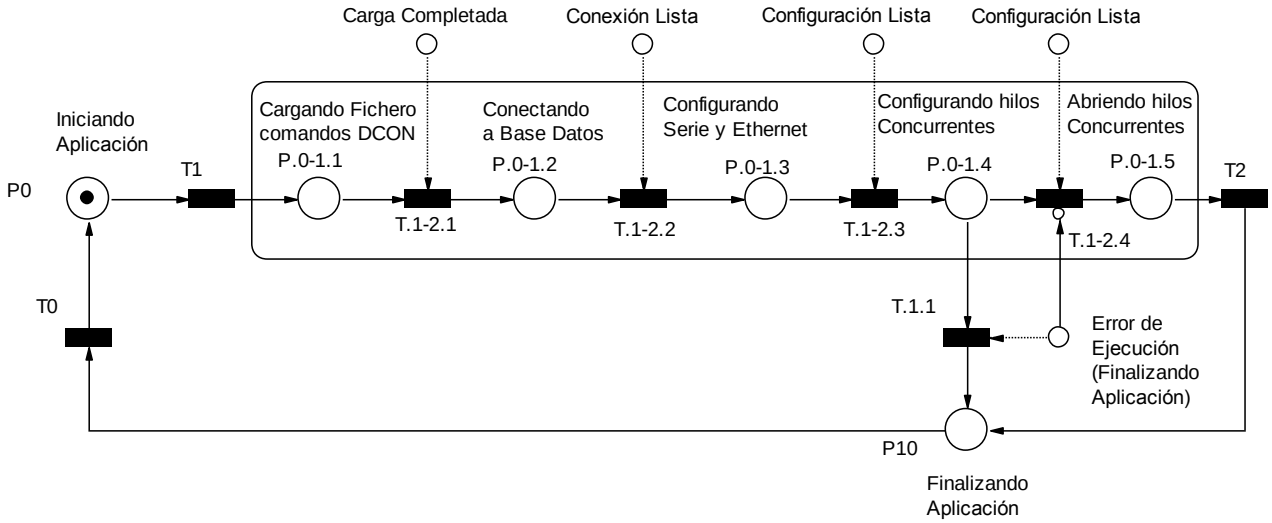
Nc: Número en la cola empleado para el escaneo anterior

ANEXO # 8: Estructura Modular del Sistema

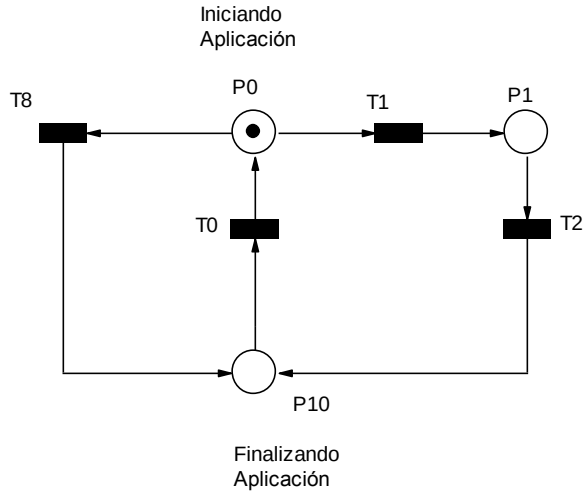


ANEXO # 9: Función Principal de GERAOpc

(a) Función Principal GERA OPC

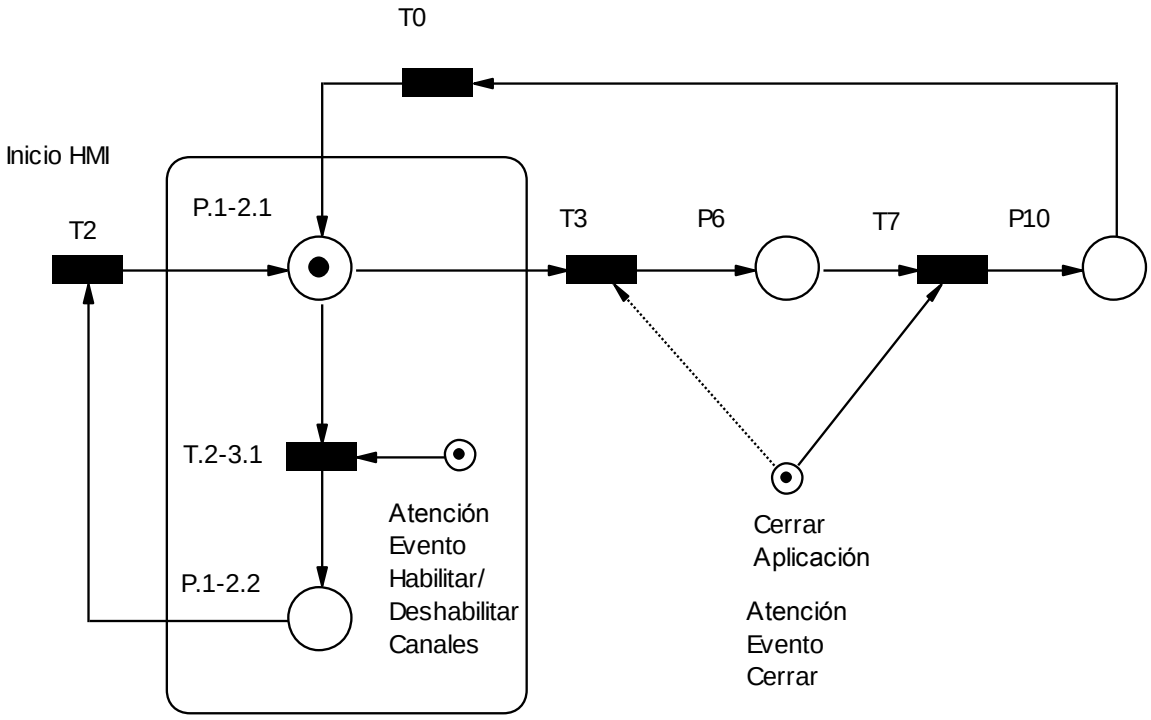


(b) Reducción de la Función Principal y correspondencia con el modelo general

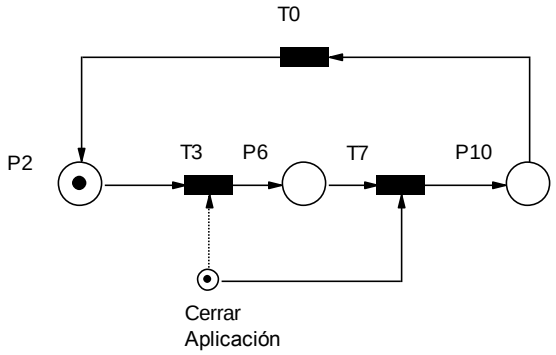


ANEXO # 10: Función Interfaz Gráfica de GERAOpc

(a) Función HMI de GERA OPC

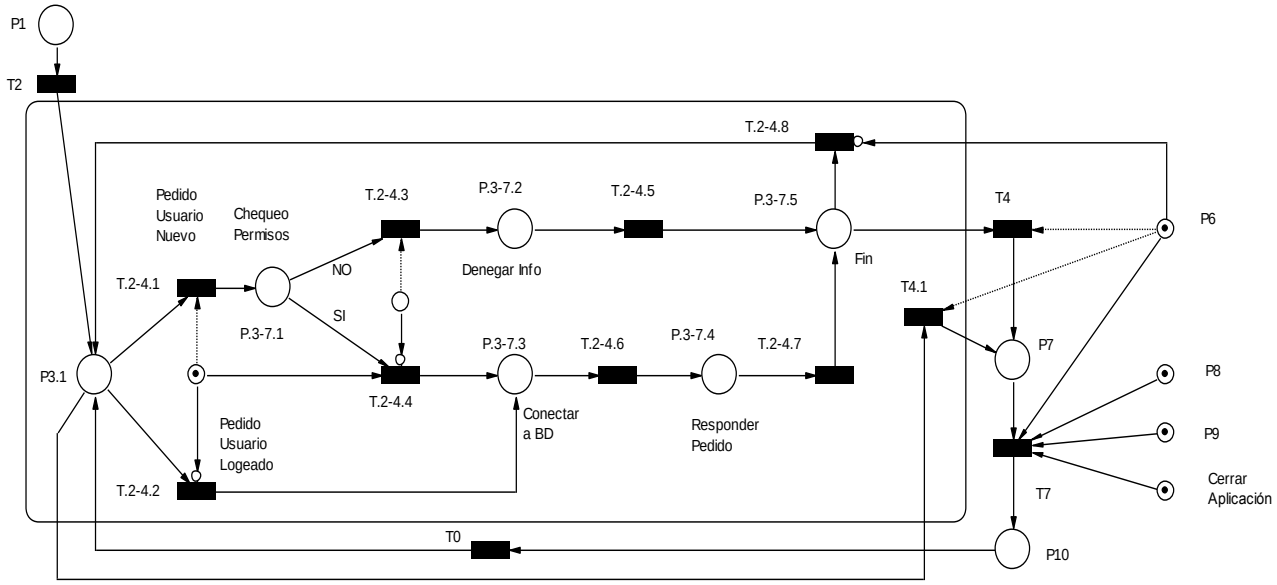


(b) Reducción de Función HMI de GERA OPC

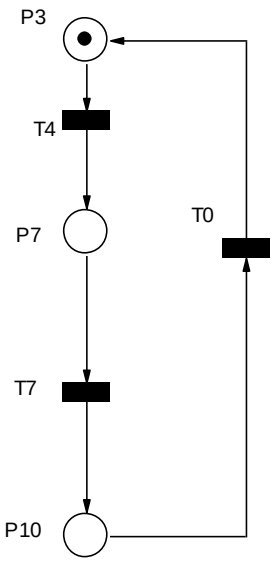


ANEXO # 11: Función Ethernet y Permisos de GERAOpc

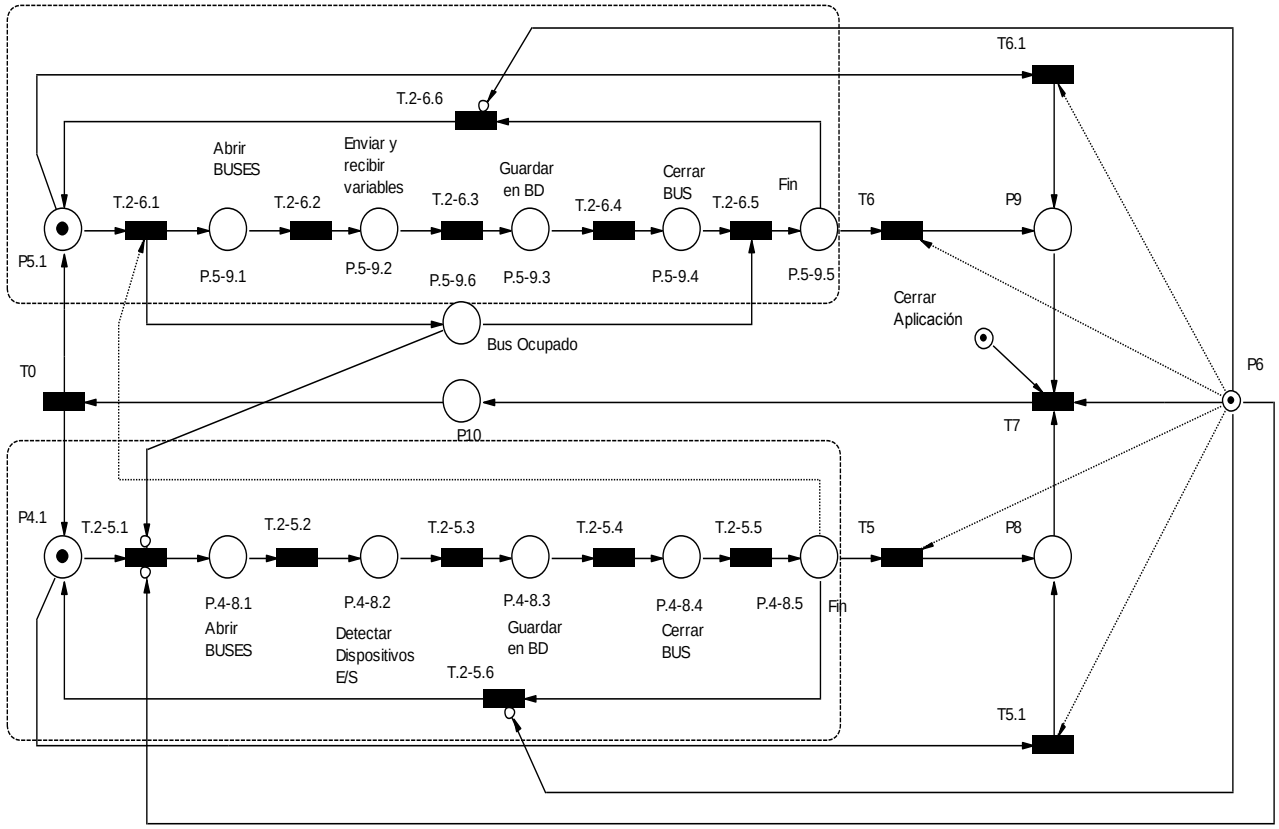
(a) Función Ethernet y Permisos de GERAOpc



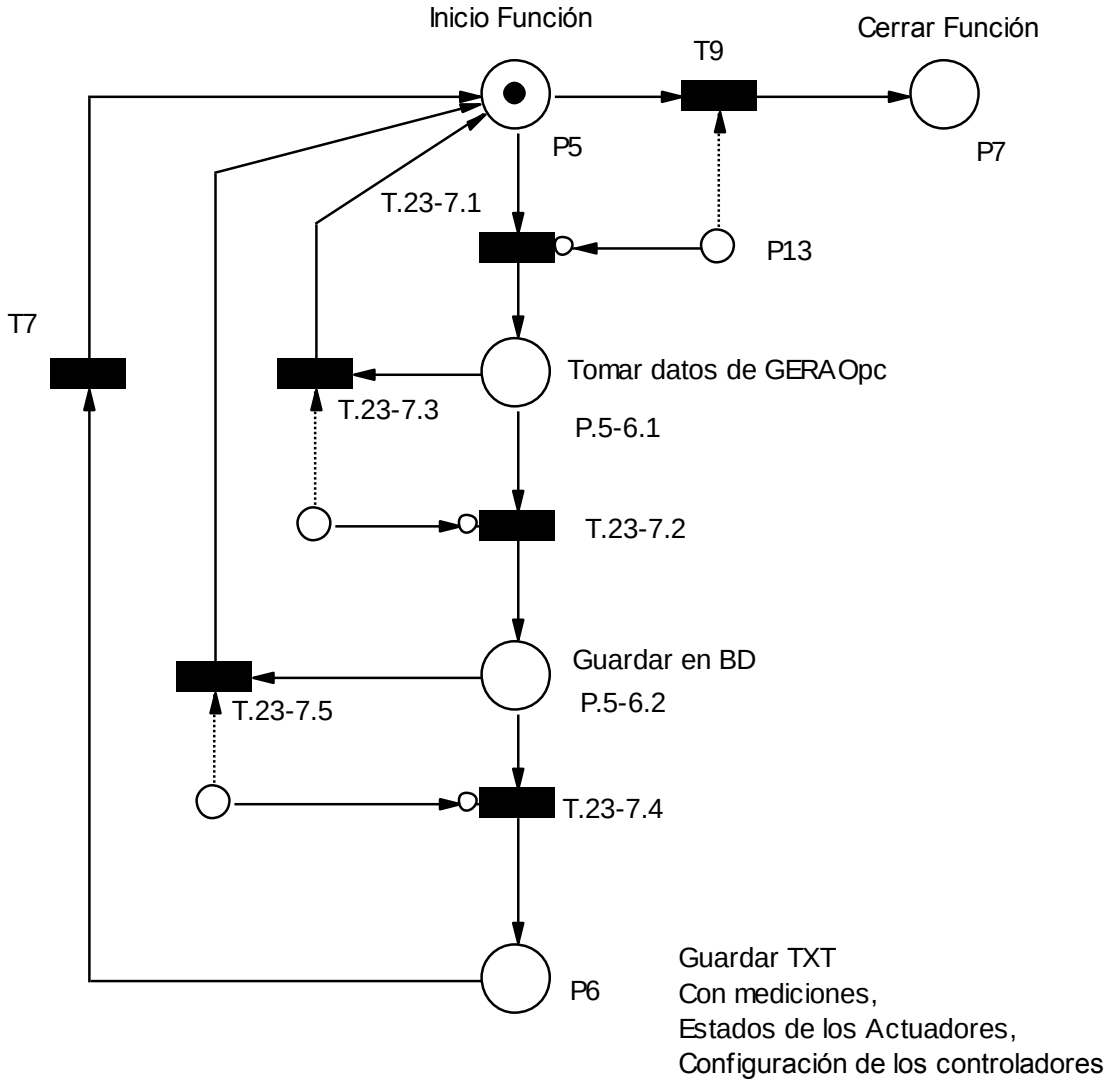
(b) Reducción de función Ethernet



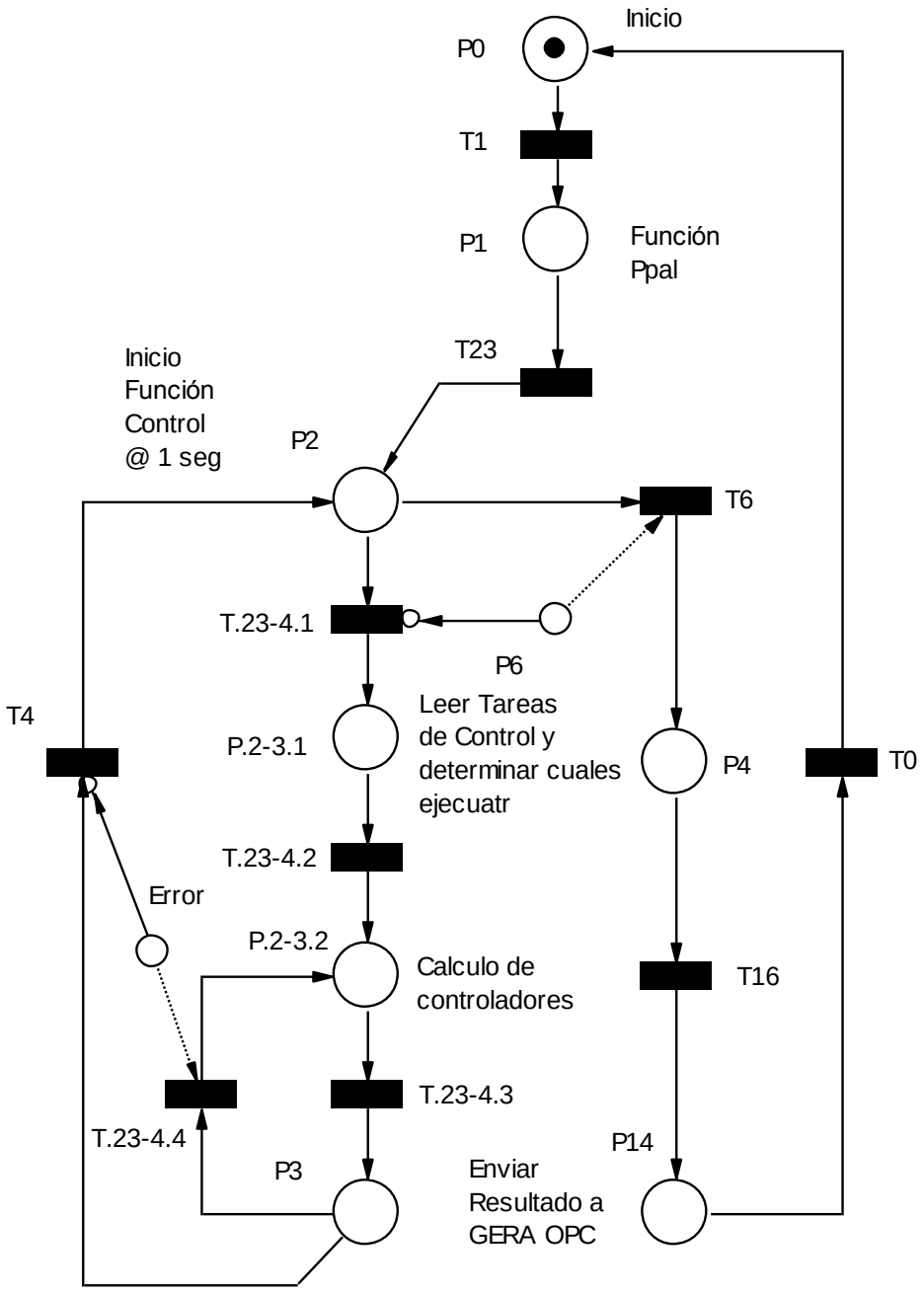
ANEXO # 12: Validación Función “Escaneo de BUS” y “Actualizar E/S”



ANEXO # 13: Función tomar datos de GERALAB

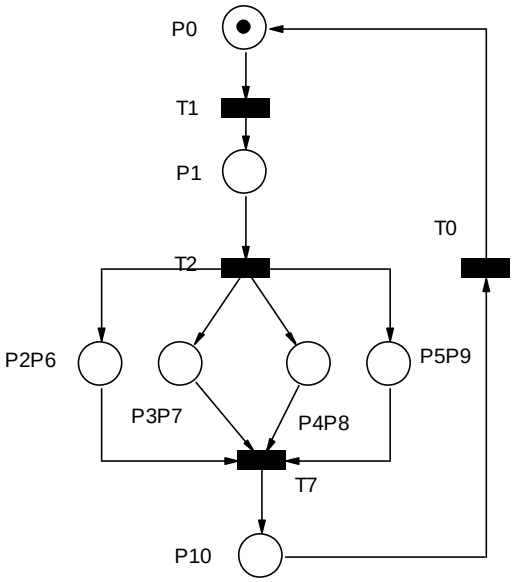


ANEXO # 14: Función control de GERALab

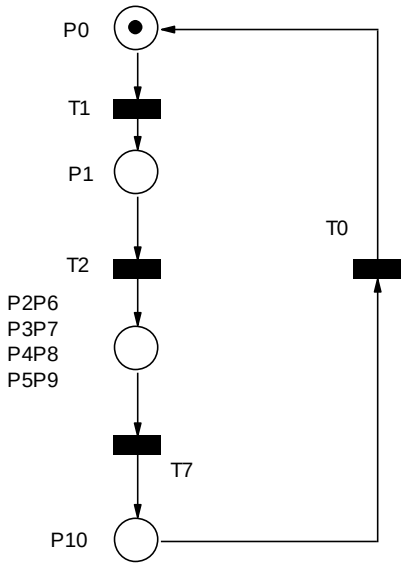


ANEXO # 15: Reducción del Modelo General de GERAOpc

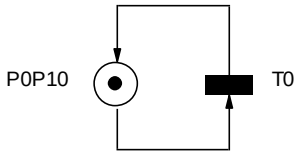
(a) Aplicación de la regla FSP



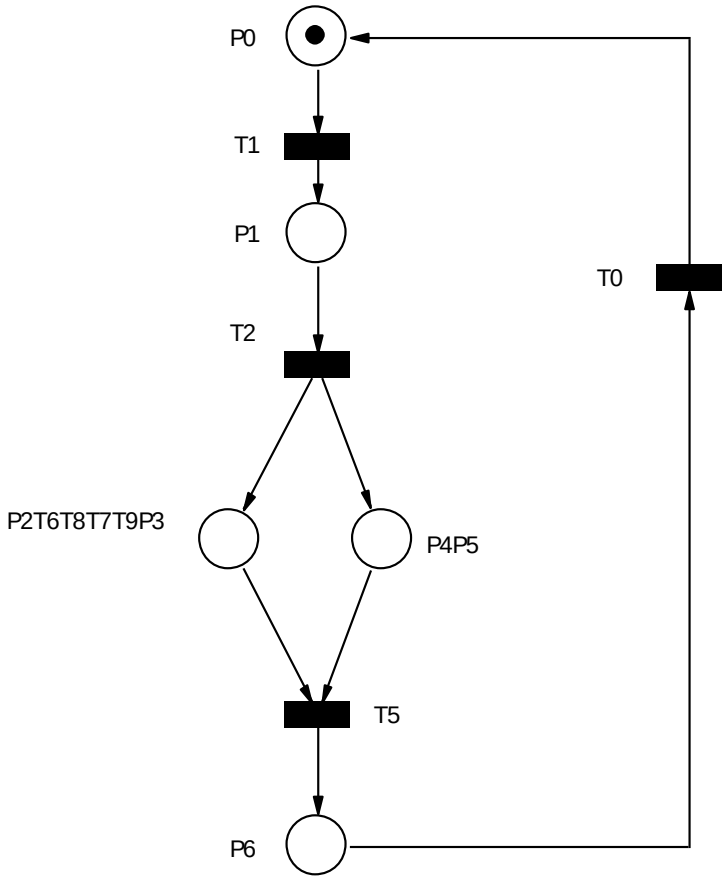
(b) Aplicación de la regla FPP



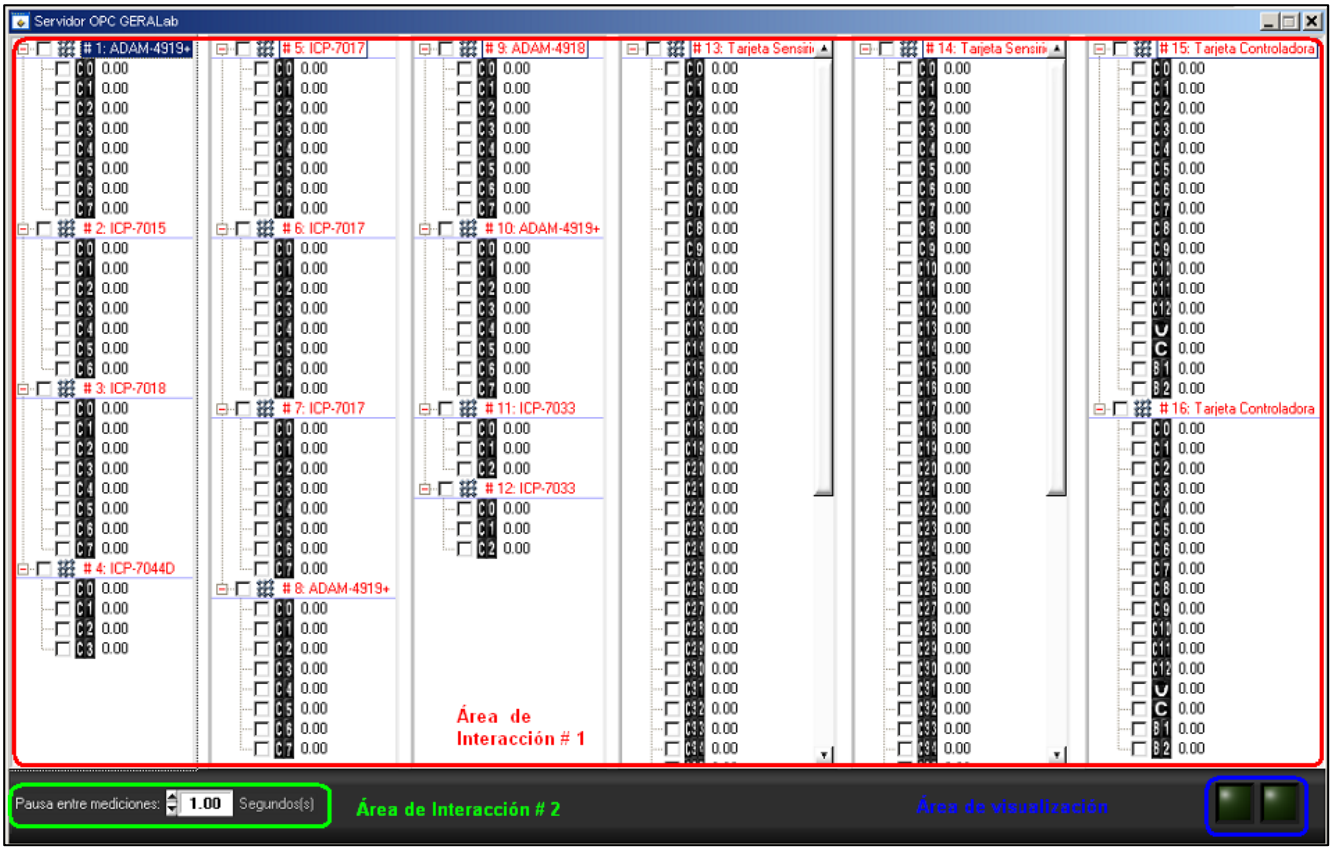
(c) Resultado final de la reducción



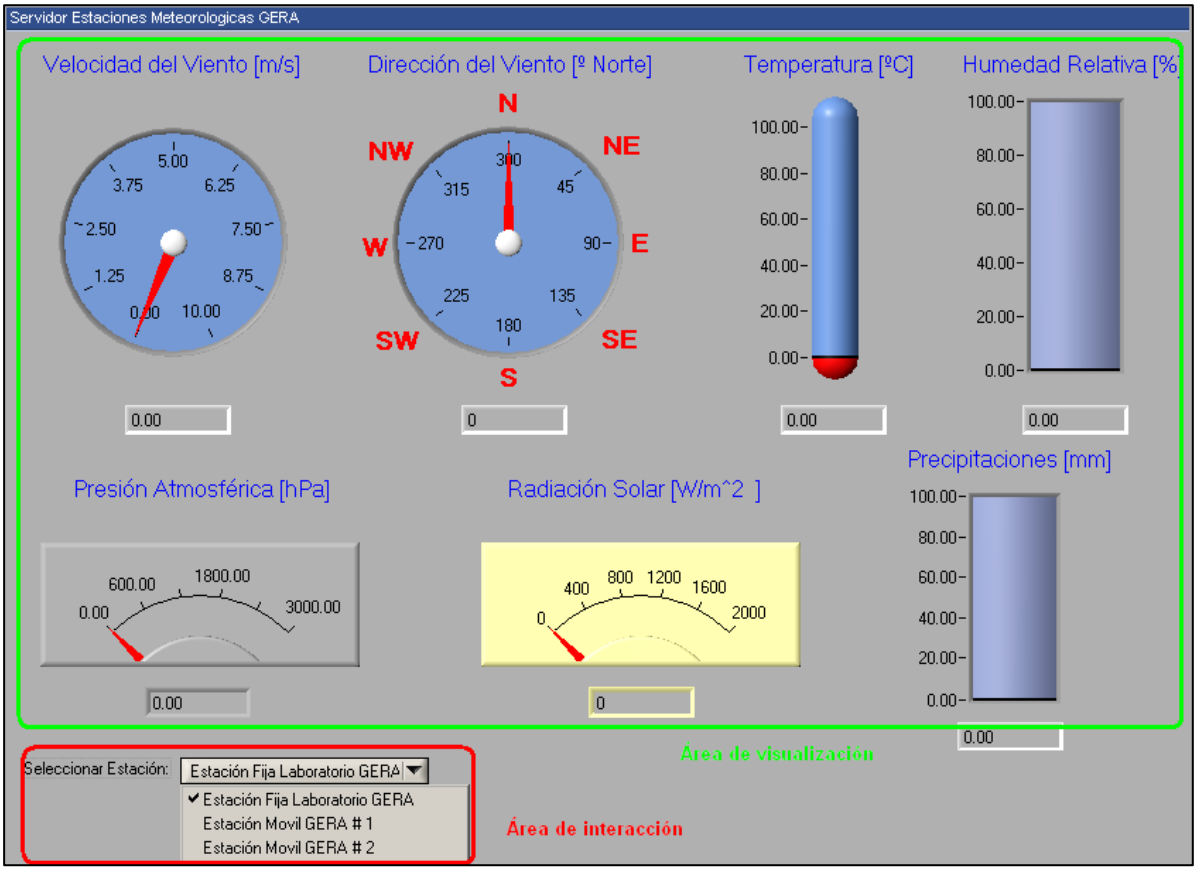
ANEXO # 16: Reducción del Modelo General de GERAStation



ANEXO # 17: Implementación del Software GERAOPC

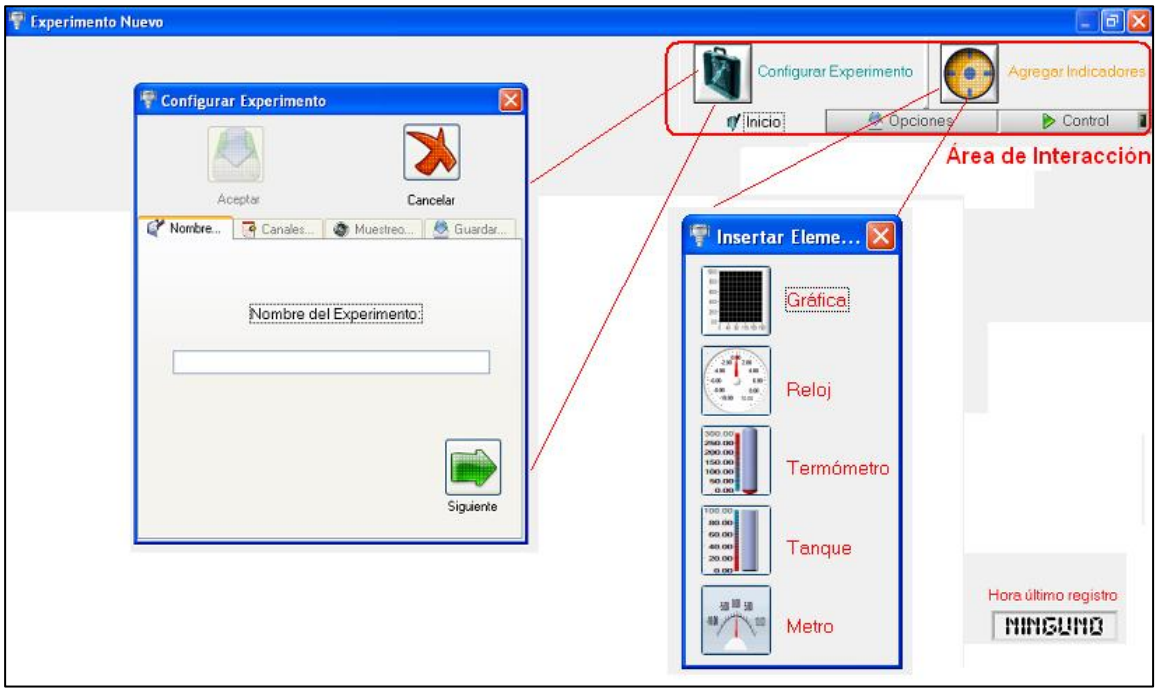


ANEXO # 18: Implementación del Software GERAStation

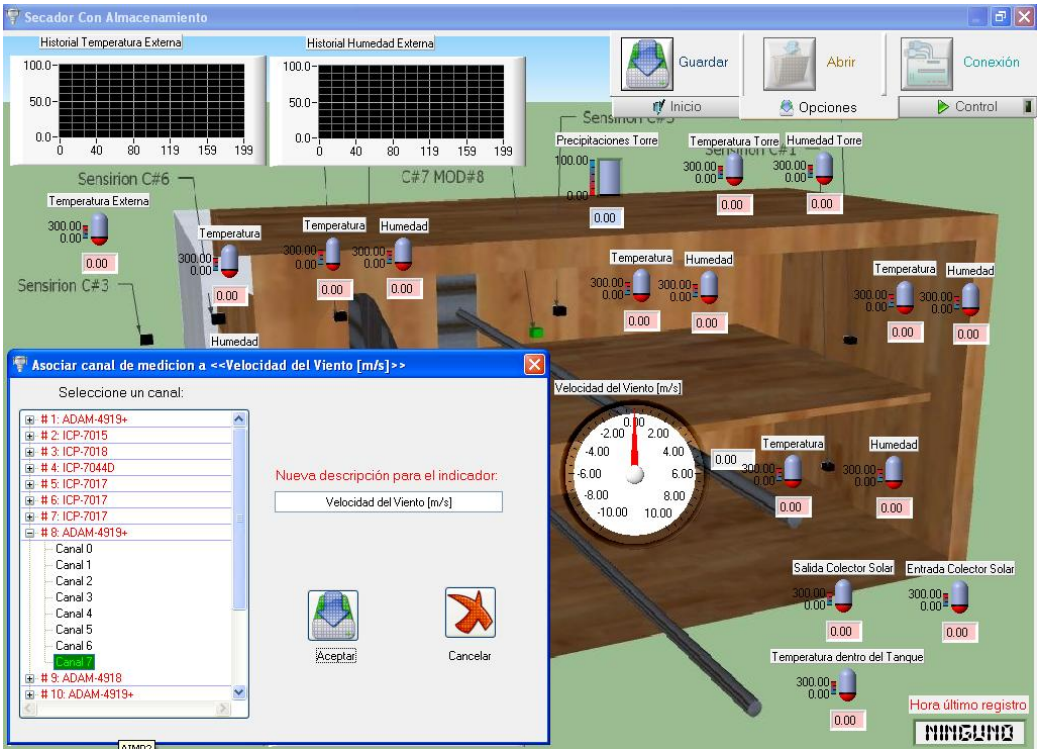


ANEXO # 19: Implementación del Software GERALab

(A) Configuración de un nuevo experimento



(B) Creación y modificación de Mímicos



ANEXO # 20: Aval

Santiago de Cuba, 25 de noviembre de 2014.

AVAL

De: Ing. Pa. Orlando Escalona Costa
Director GERA.
Universidad de Oriente.

El trabajo de tesis de fin de carrera titulado "*DESARROLLO DE UN SISTEMA SCADA FLEXIBLE PARA UN LABORATORIO DE MEDICIONES"* del autor Alfredo Pino Escalona, no es solo la culminación de la carrera del autor, sino es la culminación de una investigación de 3 años, sobre la aplicación de sistemas de medición para laboratorios en tiempo real, llevada a cabo por el autor en esta temática.

El trabajo de tesis se enmarca dentro de los temas de investigación priorizados del Grupo de Energías Renovables Aplicadas (GERA) de la facultad de Ingeniería Mecánica de la Universidad de Oriente. Siendo un gran aporte para el desarrollo de las investigaciones de este grupo, ya que con este sistema se dotó al GERA con un sistema de adquisición, monitoreo y gestión de mediciones en tiempo real, para las investigaciones relacionadas con las aplicaciones de las Energías Renovables que aquí se desarrollan.

Es necesario aclara que este es el primer y único sistema de medición en tiempo real con que cuenta la Universidad de Oriente, basado en tecnología de punta en este tema, el mismo fue desarrollado completamente por el compañero Pino Escalona, el cual participo no solo en su concepción sino que tuvo una participación decisiva en su montaje y puesta en marcha.

La dirección del GERA quiere a través de este aval, reconocer la decisiva participación del compañero Pino en la elevación de la calidad de las investigaciones del GERA, ya que con su trabajo y consagración jugó un papel decisivo en la construcción del laboratorio de ensayos solares de este grupo de investigación, único de su tipo en la región oriental del país.

Por lo que ratificamos que, el trabajo desarrollado por Alfredo Pino Escalona, puede ser calificado de excelente, demostrando competitividad, conocimiento, entrega, habilidad en el trabajo independiente y el trabajo en equipo.

Sin más,

_____
Ing. Orlando Escalona Costa
Director del GERA.