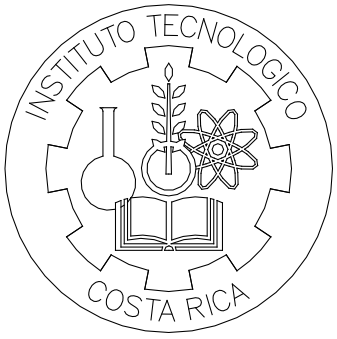
Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica



Componentes Intel de Costa Rica

Monitoreo remoto para las cámaras de choque térmico de los laboratorios de estrés de Componentes Intel de Costa Rica

Informe final para optar por el grado de Licenciatura en Ingeniería en Electrónica

Fernando Vargas Sequeira

Heredia, Noviembre 2006

Versión de 24 de noviembre de 2006

INSTITUTO TECNOLOGICO DE COSTA RICA
ESCUELA DE INGENIERIA ELECTRONICA
PROYECTO DE GRADUACIÓN
TRIBUNAL EVALUADOR

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

Miembros del Tribunal

Francisco Navarro Henríquez
Profesor lector

Anibal Coto Cortés
Profesor lector

Carlos Badilla Corrales
Profesor asesor

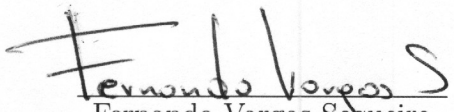
Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 20 de noviembre de 2006

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

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

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.



Fernando Vargas Sequeira
Cédula 1-1036-469

Heredia, 20 de noviembre de 2006

Resumen

Muchas aplicaciones industriales requiere operación continua o casi continua de los equipos, si esta es interrumpida puede resultar en significantes perdidas económicas. Por esta razón, es de vital importancia monitorear el estado de los principales componentes que conforman los equipos y ser capaz de detectar condiciones que puedan provocar fallas o averías en los equipos en algún momento en el futuro.

Los sistemas de mantenimiento preventivo ayudan a determinar la condición del equipo aun cuando este se encuentra funcionando en orden de detectar problemas en los componentes del equipo. El mantenimiento preventivo evalua la condición del equipo de forma periódica o continua y permite controlar los horarios de mantenimiento de forma que solo se realicen en caso de que sea necesario, obteniendose asi una mejor administración del mantenimiento.

Este documento describe en detalle el proyecto impulsado por el equipo LabSystems con el fin de mejorar el mantenimiento dado a los equipos de laboratorio de Componentes Intel de Costa Rica denominado LabMonitor, este es un prototipo de un sistema de mantenimiento predictivo que permite recolectar reportes de fallas de los distintos equipos de laboratorio y evaluar cuando es necesario aplicar el mantenimiento que se requiera.

Se expone además la solución implementada en las cámaras de choque térmico de los laboratorios de estrés de Componentes Intel de Costa Rica con el fin de resulver su integración a LabMonitor. Primeramente se expondrá el uso del Protocolo EthernetIP como solución para proveer comunicación entre el control lógico de las cámaras y los servidores de LabMonitor. Como segundo punto se explica en detalle el desarrollo de una herramienta de clasificación de condición que hace uso de la Transformada Wavelet Discreta, Algoritmos de Agrupamiento(o de Clustering) y Modelos Ocultos de Markov, con el fin de reconocer fallas a partir de las señales de vibración de los equipos. Esto representa la primer herramienta de reconocimiento de patrones desarrollada en Intel Costa Rica con esta finalidad.

Palabras Clave: Modelos Ocultos de Markov, Transformada Wavelet, Mantenimiento Predictivo, EthernetIP, TCP/IP, Clustering

Abstract

Many industrial applications require a continous operación of equipment and machines, if this is interrupted the consequences will be an increasing economic costs and downtime of equipment. For this, is very important be aple to inspect, in a periodic form, the condition of critical paramters of equipment and machines.

Predictive Maintenance (PM) helps to evaluation of equipment condition even when this is working, the target is detect catastrophic problem before those happen. This offers the possibility of decrease in costs and maintenance time, avoiding a previous inspection of equipment. PM evaluates the condition of equipment in a periodic form and permits the control of maintenance's schedules, so that this will be execute on ly when this is necessary.

This document describes in detail a project initiated by the LabSystems team, with the objetive of find a better and smarted way to increase tool availability and improve the equipment's maintenance. This project is called LabMonitorm this is a prototype of a PM system, the architecture permits pickup failure reports from the machine and evaluate when is necessary to apply a revision to equipment , increasing the efficiency of the administration of maintenance .

The solution implemented was applied to the thermal chock chambers of the stress laboratories in QR Department. First is described the use of EthernetIP protocol for the comunication with industrial equipment through an Ethenet network, and second is presented an implementation of a pattern recognition tool used for the classification of vibration signals of compressor of the chambers, this tool is based on Wavelet Discrete Transform for characteristic extraction and Hidden Markov Models for classification of signals.

KeyWords: Hidden Markov Models, Wavelet Transform, Predictive Maintenance, EthernetIP, TCP/IP, Clustering

A mami y papi

Agradecimientos

Primeramente quiero agradecer a mis padres, los cuales sin duda fueron los responsables de que pudiera lograr la meta que me impuse hace tanto tiempo, sin su apoyo y cooperación hubiera sido imposible lograrlo, les debo todo lo que tengo ahora.

A mis compañeros en la Escuela de Electrónica Jorge, Alexander Singh, Alexander Guzman, Dany, Leonardo, Fabian, Juan Manuel, Yamileth, Rolando, Lester, Ricardo y Blanca Rosa, que sin duda me han brindado todo el apoyo, ayuda, amistad y paciencia durante todo este tiempo, gracias.

Al profesor Pablo Alvarado gracias por todo el apoyo brindado, sin duda este proyecto y muchos otros no hubieran sido alcanzados sin toda la ayuda y recomendaciones hechas. Al profesor Alfonso Chacón gracias por la amistad y apoyo brindado durante todo este tiempo.

En la Escuela de Matemática a los profesores Julio Rodriguez, Sharay Meneses, Alcides Astorga, Sandra Schmidt y Silvia Calderón (gracia por enseñarme a integrar tan bien) gracias por toda la formación que fue invaluable para mi paso por la Escuela de Electrónica.

A mis compañeros en la Escuela de Matemática Ivonne, Grevin , Maureen, Benjamín y Gisella por toda la ayuda y amistad durante tanto tiempo.

En Intel Jose Araya (gracias por la oportunidad), Aurelia, Holger, Gilbert, Giovanni, Zahner, Olman, Franklin, Andrea y Mariant, por toda la ayuda brindada para el desarrollo del proyecto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la industria moderna un componente crucial del proceso de mantenimiento es un eficiente y confiable sistema de monitoreo para los distintos equipos que forman parte de los procesos industriales, lo cual permite tomar acciones correctivas cuando se presentan fallas inminentes en los equipos.

Tradicionalmente el mantenimiento que se le brinda a los equipos es realizado de forma periódica por parte del personal de mantenimiento y muchas veces requiere recurrir a inhabilitar los equipos para ejecutar la labor de mantenimiento completa. Las técnicas de mantenimiento predictivo ayudan a determinar la condición de falla de los equipos en servicio con el propositivo de predecir cuando el mantenimiento debe ser realizado. Esta estrategia ofrece reducciones en los costos y en los tiempos de inhabilitación porque los procesos de mantenimiento son realizados solamente cuando es requiere.

En este capítulo se describe la iniciativa que ha tomado el personal del grupo LabSystems en Componentes Intel de Costa Rica por desarrollar un sistema de mantenimiento predictivo que permita mejorar el mantenimiento que se le brinda a los equipos de los laboratorios de la planta en Belén y además las razones que llevaron a plantear el proyecto "Monitoreo remoto para las cámaras de choque térmico de los laboratorios de estrés de Componentes Intel de Costa Rica"

1.1 Problema existente e importancia de su solución

1.1.1 Descripción del proceso en el que se enmarca el problema

El Departamento de Calidad y Fiabilidad (Q&R) de Componentes Intel de Costa Rica tiene la tarea de ejecutar y evaluar pruebas de rendimiento y detección de fallas sobre

los lotes de producto (microprocesadores y chipsets), esto tiene como meta asegurar la calidad del producto que llega a los clientes y corregir los errores producidos en el proceso de manufactura. Para realizar esta labor Q&R, posee laboratorios y equipo especializado, en donde el producto es sometido a pruebas de rendimiento, estrés, durabilidad y carga, entre otras.

La administración de los laboratorios y sus equipos están a cargo del grupo LabSystems, el cual pertenece a la estructura de Q&R. Este grupo se encarga de la reparación, mantenimiento, control de accesos, inventarios, compra de repuestos y otras labores relacionadas con los laboratorios.

Con el fin de mejorar la calidad de la atención que se le brinda a los equipos y a sus usuarios, LabSystems ha iniciado un esfuerzo para automatizar el monitoreo y diagnóstico de fallas en los equipos de laboratorio de Q&R.

El mantenimiento que se les brinda actualmente a los equipos presenta los siguientes efectos:

1. Periodos prolongados de inactividad en los equipos, debido al tiempo utilizado para diagnosticar y reparar las fallas.
2. Incapacidad de evitar mayores daños a los equipos, por no detectar a tiempo fallas graves.
3. Administración inadecuada del inventario de repuestos, debido a que los peores periodos de reparación están asociados con la falta del repuesto.
4. Poca certeza del estado de vida de los equipos, al no poder evaluarlos de forma periódica, constante y completa.

El objetivo que se pretende lograr es implementar un sistema de mantenimiento predictivo, por medio del cual se pueda detectar fallas en los equipos mientras estos se encuentran en serivio y que se puedan tomar acciones prontas que eviten mayores daños y aislar la falla en el momento en el que es requerido sin tener que pasar por el periodo de mantenimiento del equipo.

Para esto se ha iniciado el desarrollo de herramientas de que permitan evaluar la condición de los equipos aun cuando estos se encuentran en funcionamiento y que generen reportes periodicos de su condición, además se busca poder obtener los reportes de fallas que son generados por los mismos equipos, automatizar su evaluación de forma que se de aviso en el momento en el que se presenta el problema.

En la figura 1.1 se presenta el esquema del sistema de mantenimiento predictivo. El proyecto ha tomado el nombre de LabMonitor. Su arquitectura incluye una base de datos

MySQL, en donde los reportes de fallas para los distintos equipos son almacenados y están a disposición para ser presentados en forma de reportes para los usuarios. Adicionalmente en los casos que así lo ameriten, los reportes son enviados a los respectivos encargados de laboratorio y personal de LabSystems, por medio de un correo eléctronico u otro medio que así se disponga. Junto con esto el sistema cuenta con monitoreo en paralelo para evaluar distintos equipos al mismo tiempo. Finalmente se cuenta con una interfaz de usuario que permite el manejo de la información de la base de datos del sistema de acuerdo a lo que el usuario necesite.

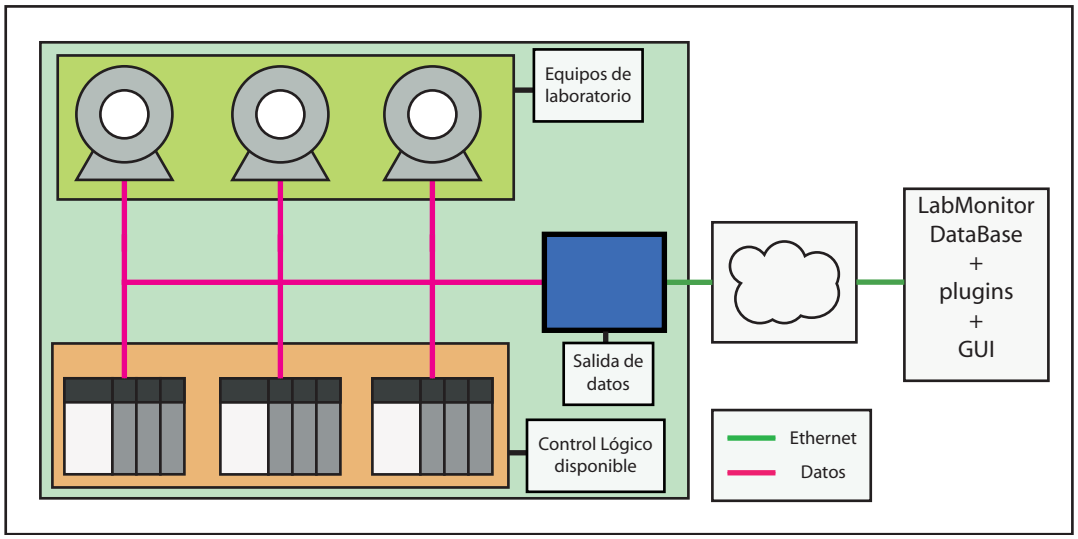


Figura 1.1: Esquema del sistema de mantenimiento predictivo LabMonitor.

1.1.2 Descripción del problema a resolver

El sistema de LabMonitor implementado requiere de un proceso de integración progresivo de los distintos equipos con los que se cuenta en los laboratorios. Este proceso debe de estar sujeto a las siguientes pautas:

1. Obtener los reportes de fallas de aquellos equipos que tienen la capacidad de generarlos e integrarlos en la base de datos de LabMonitor.

2. Implementar las medidas necesarias para generar los reportes en aquellos equipos que no lo hacen por sí mismos y que sea posible, esto ultimo debido a las limitaciones que imponen las garantías de los equipos por parte de los fabricantes y los riesgos que puede provocar modificaciones sobre su funcionamiento.

3. Brindar los medios de comunicación necesarios a aquellos equipos que lo requieran, de forma que puedan enviar sus reportes.

Como uno de los puntos iniciales de este proceso de integración se han tomado las cámaras de choque térmico [1] localizadas en los laboratorios de estrés de Q&R,las cuales se encargan de ejecutar pruebas de temperatura con cambios extremos sobre el producto.

Estas cámaras tienen la capacidad de generar reportes de alarmas a partir de los sensores que tiene incorporados, pero no poseen la capacidad de transmitir sus reportes remotamente, solo de forma local por medio de un panel táctil que funciona como terminal de visualización de datos. Junto con esto es necesario realizar un monitoreo constante principalmente sobre los compresores de su sistema de refrigeración que son un punto de vital importancia para el funcionamiento de las cámaras.

Por esto, el problema consiste en la incapacidad que poseen las cámaras de choque térmico de reportar el estado de sus componentes más importantes y los reportes de alarmas que genera, por lo que no es posible integrarlas de forma activa al sistema de mantenimiento predictivo.

Con la integración de las cámaras de estrés térmico al sistema LabMonitor se busca obtener los siguientes beneficios:

1. Optimización de los horarios de mantenimiento, reduciendo los estados de receso de las cámaras.
2. Disminuir el tiempo de reparación de las cámaras, al aislar las fallas de forma precisa, eliminando el tiempo de evaluación y diagnóstico.
3. Monitoreo continuo (24/7) y en tiempo real, permitiendo la recolección de datos para el diagnóstico de los equipos.
4. Mayor control sobre el flujo de procesos en los laboratorios, modificándolo segun sea el estado predicho por las herramientas.

1.2 Solución seleccionada

La integración de las cámaras de choque térmico en el sistema de mantenimiento predictivo LabMonitor debe cumplir con los siguientes requerimientos:

[1] Debido a las políticas corporativas de la empresa no es posible proporcionar el modelo de ningún tipo de equipo de la planta, por lo que de ahora en adelante se hará alusión a los equipos como *cámaras de choque térmico*, tampoco se hará referencia alguna a manuales, reportes, artículos ni ningún otro tipo de documento que haga alusión al modelo de los equipos.

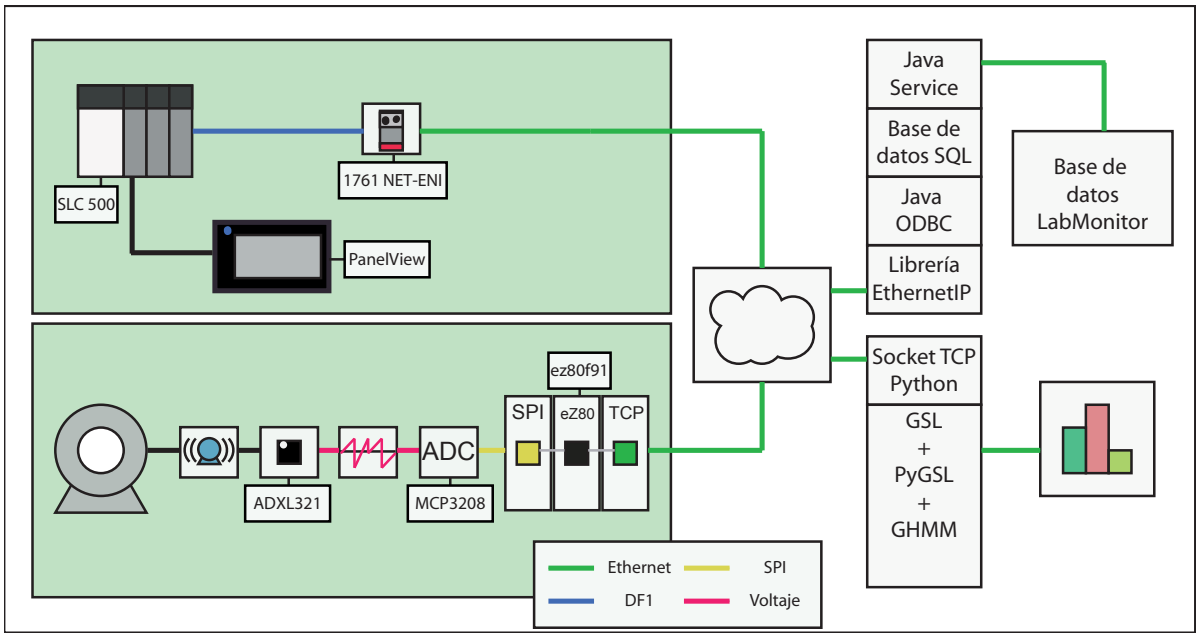


Figura 1.2: Diagrama de bloques de la solución implementada.

1. Realizar las modificaciones necesarias a los equipos de laboratorio que permitan el monitorio continuo de los equipos aun mientras estos se encuentran en servicio.

2. Desarrollar una herramienta que permita reconocer la condición de los componentes de los equipos de laboratorio.

3. Transmisión de reportes de fallas en tiempo real a través de la red ethernet de Intel.

4. Realizar los cambios necesarios en las plataformas de software que existentes en el sistema LabMonitor (base de datos e interfaz de usuario)

La integración de las cámaras de choque térmico al sistema LabMonitor comprende dos partes principales. Primero la recolección de las alarmas que son generadas debido a los sensores propios de estas, esto requiere comunicarse con el PLC de las cámaras via ethernet y seleccionar aquellas posiciones de memoria que son de interes, una vez obtenida esta información basta con generar los reportes de fallas y enviarlos a la base de datos MySQL de LabMonitor.

La segunda parte del proyecto consiste en implementar una herramienta que permita reconocer el estado de los compresores de las cámaras, esta herramienta utiliza un marco de reconocimiento basado en Modelos Ocultos de Markov (HMMs) y procesos de discretización de señales. El uso de los HMMs han sido aplicados con éxito en diversos campos como el reconocimiento de voz, clasificación de secuencias de ADN y otros. El uso de

estas teóricas esta basado en el trabajo de investigación descrito en diversas publicaciones y que ha demostrado su éxito como una herramienta en el reconocimiento del estado de equipos industriales.

En la figura 1.2 se muestra un diagrama de bloques general de la solución implementada.

La solución comprende entonces las modificaciones necesarias para lograr que los reportes de fallas de las cámaras de choque térmico sean enviados a la base de datos de Lab-Monitor por medio de la red TCP/IP disponible en los laboratorios. El segundo punto necesario para completar la solución es el desarrollo de la herramienta que permite el diagnostico del estado de los compresores a través del procesamiento de la señal de vibración que estos producen, esto incluye el hardware y software necesario para la adquisición y procesamiento de la señal.

Capítulo 2

Meta, objetivo general y objetivos específicos

La integración de las cámaras de choque térmico al sistema de mantenimiento predictivo LabMonitor tiene como propósito mejorar el mantenimiento dado a las cámaras de choque térmico propiciando el ahorro de recursos y menores tiempos de inhabilitación.

En este capítulo se detalla la meta buscada con la materialización del proyecto , la cual permitirá evaluar el impacto que tendrá el proyecto de acuerdo a las expectativas planteadas. Además se presenta el objetivo general del proyecto por medio del cual se busca dar solución al problema planteado. Finalmente se muestran los objetivos específicos que deberán ser cumplidos durante el desarrollo de la solución.

2.1 Meta

Posibilitar al sistema LabMonitor generar reportes de fallas de las cámaras de choque térmico.

2.2 Objetivo general

Monitorear remotamente las cámaras de choque térmico de los laboratorios de estrés desarrollando herramientas que permita la recolección de las alarmas producidas por las cámaras y la evaluación de la condición de sus compresores.

2.3 Objetivos específicos

1. Implementar una interfaz que permita la comunicación a través de TCP/IP de las variables que permiten el diagnóstico de fallas en las cámaras de choque térmico.

2. Desarrollar una aplicación que genere reportes de alarmas a partir de la evaluación de las variables generadas por el PLC y las integre a la base de datos de labmonitor.

3. Desarrollar un sistema de adquisición de datos que permita la transmisión de muestras de una señal de vibración a través de una red TCP/IP.

4. Desarrollar una aplicación que permita la conversión de una señal continua en una secuencia discreta.

5. Desarollar una aplicación que permita el entrenamiento de Modelos Ocultos de Markov a partir de secuencias de entrenamiento de estados conocidos y que permita el reconocimiento de secuencias de evaluación.

Capítulo 3

Marco teórico

Como se mencionó en el capítulo 1 las cámaras de choque térmico son de los primeros equipos de laboratorio que son integrados al sistema LabMonitor. Este proceso requiere el uso de distintas tecnologías de red, lenguajes de programación y teorías de reconocimiento de patrones y discretización, por esto en este capítulo se discutirán y analizarán los principales principios y teorías utilizadas para el desarrollo del proyecto, entre los temas cubiertos se encuentran TCP/IP, CIP y SPI, como tecnologías de red, también se mostrará la teoría fundamental referente al proceso de clasificación lo que incluye Transformada Wavelet Discreta, Quantización de vectores y algoritmo LBG, Mapeos de Sammon para visualización de datos de altas dimensiones, y finalmente los Modelos Ocultos de Markov y su uso para reconocimiento y clasificación de secuencias. Este capítulo no busca ser una referencia sobre estos temas sino más bien ser un pequeño vistazo sobre las posibilidades que ofrecen las diversas teorías y tecnologías usadas en la solución del proyecto, así como herramientas de software desarrolladas con el proposito de utilizar estas teorías.

3.1 Descripción del proceso a mejorar

3.1.1 Filtrado por estrés ambiental

El filtrado por estrés ambiental (ESS) es un medio de evaluación para dispositivos electrónicos, que expone defectos que no pueden ser detectados por medios visuales o por pruebas eléctricas. Estos defectos son típicamente relacionados con partes defectuosas y la mano de obra que interviene en su fabricación. ESS permite exponer el 100% del producto a estímulos ambientales por un tiempo predeterminado con el propósito de forzar la aparición de las fallas antes de que el producto sea entregado a los clientes.

El ambiente al cual el producto va a ser expuesto debe ser previamente considerado durante el proceso de diseño de la prueba. Para obtener un filtrado efectivo, se debe aplicar temperaturas extremas lo mas distanciadas que sea posible, tomando en cuenta los limites de temperatura del producto. El rango mínimo entre los niveles máximo y mínimo de temperatura debe ser cercano a los 100 °C. La tasa de cambio de temperatura del aire en la cámara debe oscilar entre los 5 °C y los 20 °C por minuto, con un énfasis especial en la uniformidad de la temperatura a través de todo el producto, lograda por una correcta circulación del aire.

Un correcto monitoreo del ESS ayudará a determinar el número de ciclos de temperatura al que el producto debe de ser expuesto, de modo que en el nivel optimo se alcance un total de cero fallas, lo que representara la ventana de filtrado mas adecuada. El éxito del ESS depende de que tan bien los resultados sean monitoreados. Determinar la causa principal de una falla es esencial para el buen desempeño de la prueba. Si la falla es detectada entonces puede ser corregida.

3.1.2 Tipos de cámaras de estrés ambiental

Existen tres tipos principales de cámaras utilizadas para aplicar el ESS. La selección es dictada por la transición de temperatura requerida:

1. Menos de 10°C por minuto, diseño tradicional.
2. 10°C por minuto a menos de 20°C por minuto, Evaporador aislado.
3. 20°C por minuto y más, cámara de choque termal aire-aire.

Diseño Tradicional

En el diseño tradicional la espiral de enfriamiento se encuentra dentro de la cámara propiamente y se utiliza un ventilador o soplador para hacer circular el aire a través de la cámara y por la espiral para enfriarlo. Durante el ciclo termal, el evaporador altera su temperatura de acuerdo al cambio en la cámara y a las necesidades del ciclo, por esto el evaporador provoca una carga en el sistema de refrigeración, lo que significa que este debe brindar funcionamiento adicional para mantener la temperatura del evaporador. Debido a que el tamaño y el peso del evaporador son proporcionales a la potencia de los compresores de refrigeración, existe un límite práctico a que tan rápido puede cambiar el ciclo de temperatura en la cámara. Entre mayor la potencia de refrigeración requerida, mas grandes y pesados son los evaporadores requeridos.

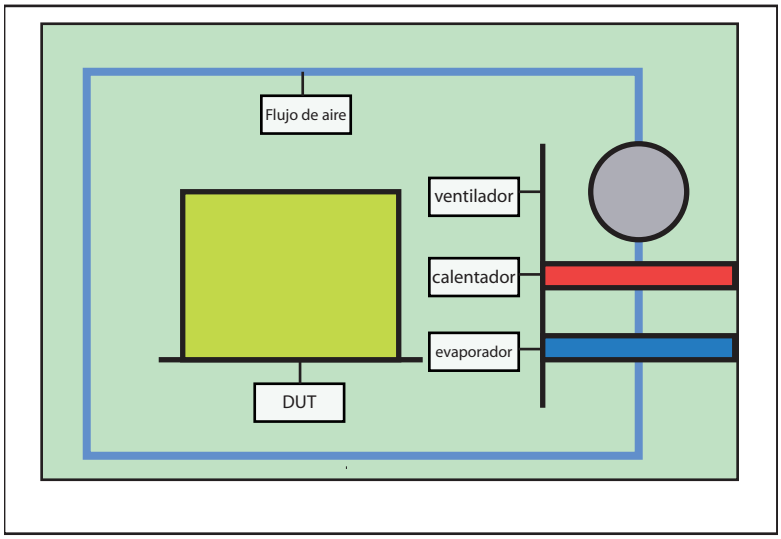


Figura 3.1: Diseño de cámara tradicional.

Evaporador aislado

El diseño de evaporador aislado remueve el evaporador como una carga para el sistema de refrigeración de la cámara. El evaporador esta aislado de la cámara en un compartimiento y es mantenido frío durante el ciclo completo de temperatura.

Una serie de amortiguadores son operados por un controlador de temperatura, abriéndolos durante el ciclo frío y cerrándolos durante el caliente, censando la cantidad de aire frío necesario para alcanzar o mantener la temperatura requerida. Debido a que el evaporador siempre se mantiene frío, este no cambia de temperatura dentro del ciclo de funcionamiento, como lo hace el resto de la cámara, de modo que no representa una carga para el sistema de refrigeración. En teoría una potencia ilimitada de refrigeración puede ser aplicada a este diseño junto con evaporadores de tamaño apropiado.

Cámara de choque termal aire-aire

El diseño de cámara de choque termal aire-aire provee la ventaja de un evaporador aislado mientras que supera el efecto del retraso de la respuesta del sistema de refrigeración durante rápidos ciclos de temperatura y al mismo tiempo reduciendo a sobre temperatura que debe ser enfriada o calentada añadida al producto bajo prueba (DUT).

El diseño aire-aire utiliza cámaras preacondicionadas en frío y caliente, transfiriendo entonces los DUTs rápidamente entre estas cumpliendo así el ciclo termal. Este diseño permite además. Este diseño permite además sobre acondicionamiento de las cámaras li-

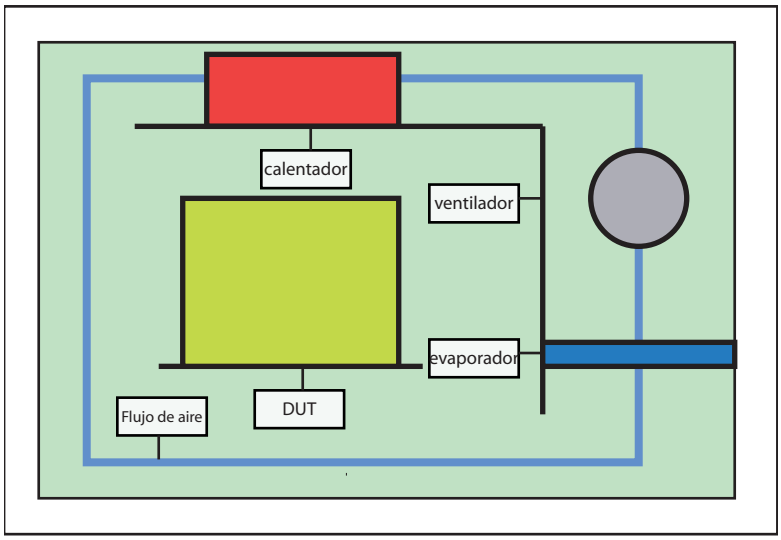


Figura 3.2: Diseño de cámara de evaporador aislado.

geramente mayor a los extremos de temperatura requeridos para acelerar la velocidad a la cual el DUT logra su temperatura meta. Debido a que la ESS requiere una transferencia de temperatura a 20°C por minuto, la cámara aire-aire es el diseño preferido.

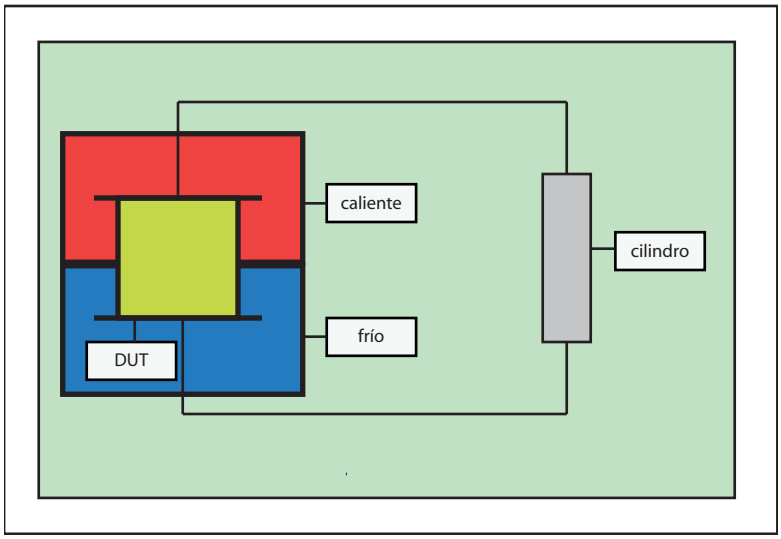


Figura 3.3: Diseño de cámara aire-aire.

3.2 Descripción de los principales principios físicos, de software y/o electrónicos relacionados con la solución del problema

3.2.1 El modelo de capas TCP/IP

Para poder obtener una visión más clara de la forma en que se constituyen las redes, su funcionamiento y componentes, se adopto la ideología de dividir su estructura en un modelo de capas, en donde cada una de estas describe un aspecto particular. Esto permite que el problema de la comunicación en una red se divida en problemas más pequeños y fáciles de manejar, lo que a la vez simplifica el aprendizaje.

El Modelo de referencia OSI

El modelo de referencia OSI es un marco que se puede utilizar para comprender cómo viaja la información a través de una red. El modelo de referencia OSI explica de qué manera los paquetes de datos viajan a través de varias capas a otro dispositivo de una red, aún cuando el remitente y el destinatario se encuentren en distintas redes con distintos medios de red.

En el modelo de referencia OSI hay siete capas numeradas. En la figura 3.4, se muestra el modelo de referencia OSI. Las capas que conforman el modelo son las siguientes:

1. Capa física: Se encarga de la descripción de los aspectos de cableado, conectores, voltajes y todo aquello que involucra el medio de red a través del cual viajan los datos.

2. Acceso al medio: Describe la forma en la que las capas superiores del modelo acceden al medio de red. Esta capa provee los medios para que el dispositivo pueda tomar control sobre el medio de red e iniciar la transmisión de los datos según lo requiera.

3. Red: Provee direccionamiento lógico dentro de la red, dando un medio de selección de rutas para la entrega de los paquetes.

4. Transporte: Describe los protocolos utilizados para controlar la transmisión de paquetes entre el emisor y el receptor. En esta capa los datos son divididos en partes antes de ser enviados. Además provee medios que garantizan la correcta entrega de los datos, así como la integridad de los mismos.

5. Sesión: Controla el inicio, permanencia y cierre de la comunicación entre el emisor y el receptor.

6. Presentación: Controla aspectos relacionados con la forma y estructura de los datos, de modo que tanto el receptor como el emisor puedan entender los datos que se están enviando.

7. Aplicación: Es la capa más alta del modelo y brinda servicios al usuario de transmisión de datos, correo electrónico, consultas web y otros.

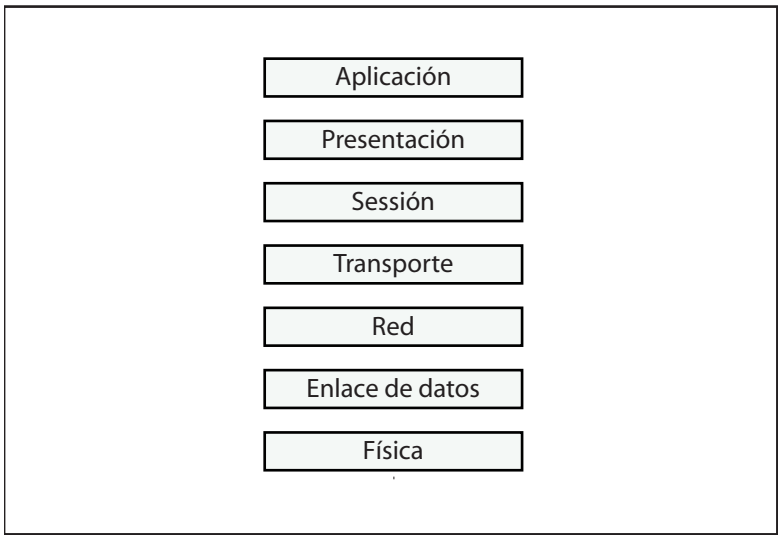


Figura 3.4: Diagrama de bloques del modelo de referencia OSI.

Las capas del modelo de referencia OSI tienen las siguientes características:

1. Cada capa en el emisor se comunica con su equivalente en el receptor, esto se conoce como comunicación de par en par.

2. Durante el proceso de comunicación cada capa comparte información con su capa equivalente, llamda unidades de datos de protocolo (PDU), cada capa en el modelo de referencia OSI tiene un PDU asociado.

3. Las capas inferiores dan servicio a las superiores, de forma que cada capa envía sus PDU a la capa inferior para que sean manejados, hasta llegar a la capa física.

4. Cada capa tiene asociado un grupo de protocolos que reciden y funcionan en esa capa, el número y función de los protocolos puede variar, pero tienen una naturaleza similar.

El modelo de capas TCP/IP

El modelo de capas TCP/IP es un estándar abierto que se utiliza ampliamente hoy en día. El modelo tiene cuatro capas: aplicación, transporte, internet y acceso a la red. Aunque algunas de las capas del modelo TCP/IP tienen el mismo nombre que las capas del modelo OSI, las capas de ambos modelos no se corresponden de manera exacta. Lo más notable es que la capa de aplicación posee funciones diferentes en cada modelo. La figura 3.5 muestra un diagrama de las capas del modelo TCP/IP.

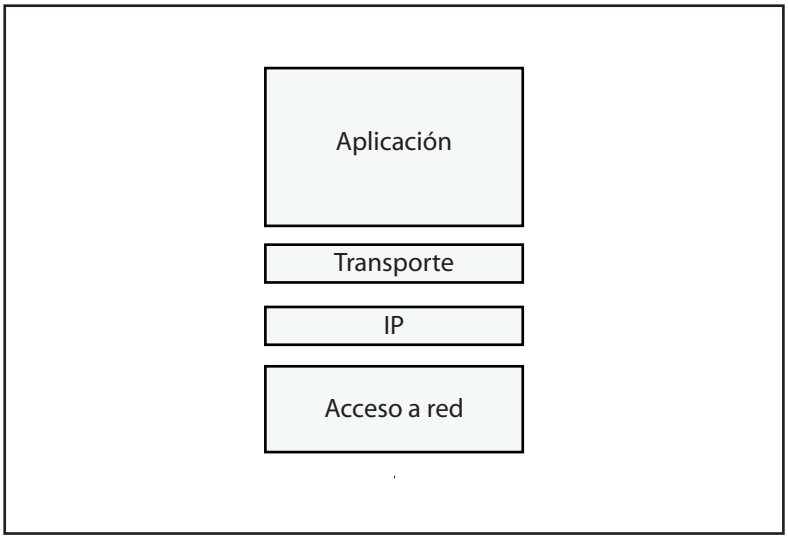


Figura 3.5: Diagrama de bloques del modelo TCP/IP.

La capa de aplicación en el modelo TCP/IP incluye los aspectos de las capas de sesión y presentación del modelo OSI. Así crearon una capa de aplicación que maneja aspectos de representación, codificación y control de diálogo. En esta capa residen protocolos como HTTP, SMTP, FTP y otros más.

La capa de transporte se encarga de los aspectos de calidad del servicio con respecto a la confiabilidad, el control del flujo y la correción de errores. En esta capa recide el protocolo TCP, el cual es un protocolo de transporte orientado a la conexión, que brinda servicios de transporte seguro, y el protocolo UDP el cual es un protocolo no orientado a conexion.

La capa internet tiene como propósito dividir los segmentos TCP en paquetes y enviarlos desde cualquier red. Los paquetes llegan a la red destino independientemente de la ruta que utilizaron para llegar allí. El protocolo específico que rige esta capa se denomina protocolo Internet (IP). Además reciden algunos otros protocolos como ARP, RARP y otros más. En esta capa se produce la determinación de la mejor ruta y la conmutación de paquetes.

La capa de acceso a red guarda relación con todos los componentes, tanto físicos como lógicos, necesarios para lograr un enlace físico. Incluye los detalles de networking, y todos los detalles de las capas física y de enlace de datos del modelo OSI. En esta capa reciden protocolos como Ethernet, y CSMA/CD.

3.2.2 Protocolo de Control e Información CIP

Protocolo EthernetIP

EthernetIP es un protocolo de aplicacion por capas para aplicaciones de automatización industrial. Esta basado en los protocolos TCP/IP y utiliza la tecnología de hardware y software de Ethernet, definiendo una capa de aplicación para configurar, acceder y controlar dispositivos de automatización industrial. EthernetIP clasifica los nodos Ethernet como tipos predefinidos de dispositivos con comportamientos especificos.

El medio físico de Ethernet, cableado y conectores que se utilizan en los computadores de oficina, impresoras y otros dispositivos periféricos, manejan una serie de protocolos de comunicación como IP, TCP y muchos otros más. Este grupo de protocolos y medios de conectividad es bien conocida en ambientes de oficina, lo que permite a los usuarios compartir archivos, acceder impresoras, enviar correos electrónicos y acceder a la internet. Las necesidades en el piso de producción son mucho más demandantes y necesitan cumplir con una serie de requisitos. En este ambiente industrial los controladores deben acceder datos de sistemas de control, servidores y dispositivos de entrada/salida. En condiciones normales, la aplicación hace esperar al usuario mientras un proceso se lleva a cabo. Por otro lado en el piso de producción, las aplicaciones dependen del tiempo como factor clave y requieren comunicaciones en tiempo real. El manejo de un brazo robotico requiere una temporización más precisa que en el caso del acceso a un archivo a un servidor de datos de una oficina.

EthernetIP soporta tasas de transmisión de datos de 10 Mbps y 100 Mbps. Normalmente se utiliza una topología de estrella utilizando dispositivos tradicionales de Ethernet como switches o hubs. el número de dispositivos conectados en una red EthernetIP depende solo del espacio de direccionamiento IP disponible.

EthernetIp es una de tres estandards de red abiertos (DeviceNet, ControlNet y EthernetIp) que utilizan como capa de aplicación el protocolo CIP (Common industrial Protocol). EthernetIP es la aplicación de CIP sobre TCP/IP y Ethernet (IEEE 802.3). Además de la automatización industrial que incluye módulos de entrada/salida, valvulas, codificadores, drivers y PLCs, el principal campo de aplicación de EthernetIP son las redes tanto de control como del nivel ejecutivo. Dentro de la familia de redes CIP, EthernetIP cubre las

aplicaciones donde un medio a alto nivel de datos necesita ser intercambiado en una red.

El protocolo CIP

El protocolo CIP (Common industrial Protocol) es un protocol orientado a objetos de punto a punto que provee conexión entre dispositivos industriales (sensores, actuadores) y dispositivos de alto nivel (controladores). CIP es independiente de la capa física y de acceso al medio que se utilice.

CIP tiene los siguientes propósitos:

1. Transporte de datos orientados a control asociados con dispositivos de entrada/salida.
2. Transporte de otro tipo de información la cual esta relacionada con el sistema controlado, como parámetros de configuración y diagnóstico.
3. Las capas inferiores dan servicio a las superiores, de forma que cada capa envía sus PDU a la capa inferior para que sean manejados, hasta llegar a la capa física.
4. Cada capa tiene asociado un grupo de protocolos que reciden y funcionan en esa capa, el número y función de los protocolos puede variar, pero tienen una naturaleza similar.

CIP hace uso de modelado abstracto de objetos para describir:

1. El conjunto de servicios de comunicación disponibles.
2. El comportamiento externo visible de un nodo CIP.
3. Un medio común por medio del cual la información en los productos CIP es accesada e intercambiada.

Un nodo CIP es modelado como una colección de objetos. Un objeto provee una representación abstracta de un componente en particular dentro de un producto; la realización de este modelo es una implementación dependiente, en otras palabras, un producto internamente mapea este modelo de objeto en una forma específica para esta implementación.

El protocolo de encapsulación CIP define un número de puerto TCP que debe de ser soportado por todos los dispositivos EthernetIP. Todos los dispositivos EthernetIP deben aceptar al menos dos conexiones en el puerto TCP 44818. Para UDP, se define el mismo puerto para la llegada de mensajes.

Tabla 3.1: Estructura del paquete CIP

Estructura	Nombre del campo	Valor
Encabezado de encapsulación	comando	Comando de encapsulación
	longitud	Longitud en bytes de la porción de datos del mensaje
	Manejador de sesión	Identificador de la sesión
	Status	Código de status
	Contexto del remitente	Información pertinente al remitente
	Opciones	Banderas de opciones
Datos específicos de comando	Datos encapsulados	Datos

Encapsulamiento de paquetes CIP

Todos los mensajes enviados, via TCP o UDP al puerto 44818, deben estar compuestos de un encabezado estructurado de 24 bytes seguido por una porción de datos opcional. El total de la longitud del mensaje (incluyendo el encabezado) debe de estar limitado a 65535 bytes. La estructura del mensaje debe ser como en la tabla 3.1.

Los componentes de la estructura son:

1. Comando (command field): Este campo describe el tipo de comando que debe ser ejecutado por el dispositivo, este debe de aceptar cualquier comando aun si este no lo soporta sin romper la sección o la conexión TCP. Un código de status indicando que un comando no soportado se ha enviado debe ser returnado al emisor del mensaje, en la siguiente sesión se hace una descripción de los principales comandos utilizados para el proyecto.

2. Longitud de datos(Length field): Este campo debe de especificar el tamaño en bytes de la porción del menasaje correspondiente a los datos. El valor del campo debe ser igual a cero en el caso de mensajes que no posean datos. El tamaño total del mensaje debe ser la suma del número contenido en el campo de longitud sumano a los 24 bytes del encabezado de encapsulación.

3. Manejo de sesión (sesión handle): El manejo de sesión debe de ser generado por el receptor y retornado al emisor en respuesta a una solicitud de registro de sesión. El emisor debe insertar este valor en todos los paquetes siguientes que se envien al receptor.

4. Campo de status(Status field): El valor del campo de status indica cuando el receptor esta capacitado de ejectuar el comando solicitado en el paquete. Un valor de cero debe indicar que se ha ejecutado con éxito el comando. En todas las solicitudes enviadas por el emisor, este campo debe ser puesto en cero. Si el receptor recibe un paquete con un valor diferente de cero en el campo de status, la solicitud debe ser ignorada por el receptor y no se genera ninguna respuesta.

5. contexto de remitente(Sender context field): El remitente del comando debe asignar el valor de este campo en el encabezado. El receptor debe retornar este valro sin modificación en su respuesta. Los comandos que no esperan una respuesta pueden ignorar este campo.

6. Opciones(Options field): El remitente de un paquete debe de poner las opciones en cero, lo que debe de ser verificado por el receptor, en caso de no cumplirse esto el paquete debe de ser descartado.

7. Datos de comando específico(Command specific data field): La estructura de este campo depende del tipo de comando. Para organizarlo se utiliza alguno o ambos de los siguientes métodos: uso de una estructura arreglada o bien uso de formato de paquete común (common packet format), descrito en la sección 3.2.2.

Aunque el encabezado no contiene información explicita para distinguir entre una solicitud o una respuesta, esta información debe ser determinada en dos maneras, la primera implicitamente por el comando y el contexto en el cual el mensaje es generado, y segundo explícitamente por los contenidos de un paquete de protocolo encapsulado en la parte de datos del mensaje.

Descripción de los comandos CIP

A continuación se da una breve descripción de los comandos CIP utilizados para la comunicación con el cliente EthernetIP:

1. Lista de servicios (List Services): Este determina cuales clases de servicios de encapsulación soporta el dispositivo con el que se desea iniciar conexión. Cada clase de servicio tiene un código de tipo único y un nombre ASCII opcional. Una clase de servicio es definido, con el código de tipo 0x100 y nombre "communications". Esta clase de servicio debe indicar que el dispositivo soporta encapsulación de paquetes CIP. Todos los dispositivos que soportan encapsulación CIP deben soportar la solicitud de lista de servicios y la clase de servicio "communications". En la 3.2 se muestra el encabezado del mensaje CIP para la solicitud de lista de servicios.

Tabla 3.2: Encabezado del mensaje CIP para la solicitud de lista de servicios

Estructura	Nombre del campo	Valor
Encabezado de encapsulación	comando	0x04
	longitud	0
	Manejador de sesión	Ignorado
	Status	0
	Contexto del remitente	0
	Opciones	0

Tabla 3.3: Encabezado del mensaje CIP para la solicitud de la lista de interfaces

Estructura	Nombre del campo	Valor
Encabezado de encapsulación	comando	0x64
	longitud	0
	Manejador de sesión	Ignorado
	Status	0
	Contexto del remitente	0
	Opciones	0

2. Lista de interfaces (List Interfaces): Este comando debe ser utilizado por el cliente para identificar interfaces de comunicación que no corresponden al protocolo CIP en el dispositivo al que se desea conectar. En la tabla 3.4 se muestra el encabezado del mensaje CIP para la solicitud de lista de interfaces.

3. Registrar sesión (Register sesión): Este comando origina la sesión con el dispositivo al que se desea conectar. El comando devuelve un número de sesión el cual se utiliza para realizar todas las subsecuentes solicitudes al dispositivo. En la tabla **??** se muestra del encabezado del mensaje CIP para el comando de registro de sesión.

Formato del common packet

El formato del common packet define un formato estandard para los paquetes que son transportados con el protocolo de encapsulación. EL formato del common packet es un mecanicsmo de propósito general diseñado para acomomdar tipos de paquetes y direcciones en el futuro. El common packet forma parte del command specific data.

El formato de common packet debe consistir de un contador de miembros (item count),

Tabla 3.4: Encabezado del mensaje CIP para la solicitud de la lista de interfaces

Estructura	**Nombre del campo**	**Valor**
Encabezado de encapsulación	comando	0x65
	longitud	4
	Manejador de sesión	0
	Status	0
	Contexto del remitente	Cualquier valor
	Opciones	0
Datos específicos de comando	Version del protocolo	0x01
	Banderas	0

Tabla 3.5: Estructura formal de este campo del mensaje

Nombre del campo	**Descripción**
Conteo de ítemes	Número de ítemes siguientes
Dirección de ítem	Dirección para el encapsulamiento
Datos de ítem	Paquete de datos encapsulado

seguido por una dirección de miembro (address item), y finalmente los datos del miembro (data item). En la tabla 3.5 se muestra la estructura formal de este campo del mensaje.

La estructura del los campos de datos y dirección de miembro deben ser como en la tabla 3.6.

Tabla 3.6: Estructura del los campos de datos y dirección

Nombre del campo	**Descripción**
Id tipo	Tipo de ítem encapsulado
Longitud	Longitud en bytes de los datos siguientes
Datos	Datos

3.2.3 Protocolo SPI (Serial Peripherial Interface)

Spi es un bus de tres líneas, sobre el cual se transmiten paquetes de información de 8 bits. Cada una de estas tres líneas porta la información entre los diferentes dispositivos conectados al bus. Cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo, por lo que este tipo de comunicación serial es full duplex. Dos de estas líneas trasfieren los datos (una en cada dirección) y la tercera línea es la del reloj. Algunos dispositivos solo pueden ser transmisores y otros solo receptores, generalmente un dispositivo que tramite datos también puede recibirlos.

Los dispositivos conectados al bus son definidos como maestros y esclavos. Un maestro es aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y control. Un esclavo es un dispositivo controlado por el maestro. Cada esclavo es controlado sobre el bus a través de una línea selectora llamada Chip Select o Select Slave, por lo tanto es esclavo es activado solo cuando esta línea es seleccionada. Generalmente una línea de selección es dedicada para cada esclavo. En un tiempo determinado, solo podrá existir un maestro sobre el bus. Cualquier dispositivo esclavo que no este seleccionado, debe deshabilitarse (ponerlo en alta impedancia) a través de la línea selectora (chip select). El bus SPI emplea un simple registro de desplazamiento para transmitir la información.

3.2.4 La Transformada Wavelet

Las transformaciones matemáticas son aplicadas a las señales para obtener alguna información que sea útil, que no es accesible por simple inspección de la señal. Una señal en el dominio del tiempo (una representación amplitud- tiempo) es generalmente la base de información a partir de la cual se parte, la señal procesada representa la aplicación de la transformación matemática para obtener el resultado que se espera. El espectro en frecuencia de una señal es básicamente una representación del contenido de componentes de frecuencia que están presentes en la señal.

Es bien conocido de la teoría de Fourier que una señal puede ser representada por medio de una suma de, posiblemente infinita, serie de senos y cosenos. Esta suma es referida también como expansión de Fourier. La gran desventaja de la expansión de fourier es que solamente tiene resolución en frecuencia y no resolución en el tiempo. Esto significa que aunque es posible conocer todas las frecuencias que están presentes en una señal, no es posible conocer el momento en que estas se presentan. Las llamadas señales no estacionarias son aquellas en donde el contenido de frecuencia de la señal cambia continuamente en el tiempo, de modo que una representación en la frecuencia como la de Fourier no representa una descripción adecuada de la señal en muchas aplicaciones. Para resolver este problema se han presentado múltiples soluciones en el pasado que han representado

aproximadamente en el dominio del tiempo y la frecuencia al mismo tiempo.

La idea principal detrás de estas representaciones en el tiempo y la frecuencia es la de cortar la señal de interés en varias partes y entonces analizar estas partes de forma separada. Es claro que analizar una señal de esta forma dará la información del contenido de frecuencias que se presenta en la señal y cuando ocurren, pero esto conduce a un problema fundamental, que es: como cortar la señal. Supóngase que se quiere saber exactamente cual es el todas las componentes de frecuencia presentes en un momento específico en el tiempo. Entonces por medio del uso de una ventana muy corta de tiempo como un pulso de Dirac se corta la señal y se hace el análisis de frecuencia, pero esto no es del todo acertado.

El problema que se tiene es que cortar la señal corresponde a la convolución de la señal con la ventana cortante. Debido a que la convolución en el dominio del tiempo es idéntica a la multiplicación en el dominio de la frecuencia y debido a que la transformada de fourier del pulso de Dirac contiene todas los posibles frecuencias, los componentes de frecuencia de la señal serán esparcidos a través de todo el eje de frecuencia. De hecho esta situación es el opuesto a la transformada de fourier estandar debido a que se tiene resolución en el tiempo pero no en la frecuencia. La transformada wavelet es la solución más reciente que se ha utilizado para resolver los problemas con la transformada de fourier. En el análisis wavelet el uso de una ventana escalable resuelve el problema del recorte de la señal. La ventana es trasladada a través de la señal y para cada posición el espectro es calculado. Entonces este proceso es repetido muchas veces con una ventana más delgada o más ancha para cada nuevo ciclo. Al final el resultado será una colección de representaciones tiempo-frecuencia de la señal, todas con diferentes resoluciónes. Debido a esto se se habla de un análisis multiresolución. En el caso de wavelets no se habla comúnmente de representaciones tiempo-frecuencia sino más bien de representaciones tiempo-escala, donde la escala es lo opuesto a la frecuencia, debido a que el termino frecuencia se reserva para el análisis de Fourier.

La transformada continua wavelet

El análisis wavelet descrito en la sección anterior es conocido como transformada wavelet continua o CWT. Más formalmente está se representa por medio de

$$\gamma(s,\tau) = \int f(t)\psi^*_{s,\tau}(t)dt \tag{3.1}$$

donde * denota la conjugación compleja. Esta ecuación muestra que una función $f(t)$ es descompuesta en un conjunto de funciones básicas $\psi_{s,\tau}(t)$, llamadas wavelets. Las varia-

bles s y τ son las nuevas dimensiones, escala y traslación, después de la transformación. Para complementar la ecuación 3.1 se muestra a continuación la transformada wavelet inversa

$$f(t) = \int \int \gamma(s,\tau)\psi_{s,\tau}(t)d\tau ds \tag{3.2}$$

Las wavelets son generadas por medio de una wavelet básica $\psi(t)$, que es llamada la madre wavelet, escalando y trasladando la función

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{3.3}$$

donde s es el factor de escala, τ es el factor de translación y el factor $s^{\frac{-1}{2}}$ es para la normalización de la energía a través de las distintas escalas.

El parámetro escala en el análisis wavelet es similar a la escala utilizada en los mapas. Como en el caso de los mapas, las altas escalas corresponde a una vista no detallada del terreno, o de la señal para el caso de los wavelets, y de bajas escalas correspondientes a vistas detalladas. De forma similar, en terminos de la frecuencia, bajas frecuencias (altas escalas) corresponden a una información golbal de la señal (que usualmente cubre toda la señal), mientras que las altas frecuencias (bajas escalas) corresponde a información detallada de una patrón oculto en la señal (que usualmente se encuentra en un periodo corto de tiempo).

El escalamiento, como una operación matemática, ya sea dilata o comprime una señal. Grandes escalas corresponden a versiones dilatadas de la señal y pequeñas escalas corresponden a versiones comprimidas de la señal.

En terminos de funciones matemáticas, si $f(t)$ es una función dada entonces $f(st)$ corresponde a una versión comprimida de $f(t)$ si $s > 1$ y a una versión expandida de $f(t)$ si $s < 1$. Sin embargo en la definición de la transformada wavelet, el termino de escalamiento s se encuentra en el denominador, y por esto lo anterior corresponde a lo opuesto para la transformada, esto es, las escalas donde $s > 1$ expande la señal mientras que las escalas $s < 1$ comprime la señal.

El cálculo de la transformada wavelet continua puede ser fácilmente comprendida a partir de su definición de la ecuación 3.1, es un proceso de escalamiento y desplazamiento, existen otros referencias que muestran ejemplos del cálculo de esta versión de la transformada

wavelet como es el caso de [19], por lo que no se aundará en esto, mas bien se discutirá el tema de la resolución de la transformada wavelet y la transformada discreta wavelet utilizada en cálculos computacionales.

Resolución en el tiempo y la frecuencia

La figura 3.6 se utiliza comunmente para mostrar las propiedades de resolución que presenta la transformada wavelet y como debe de ser interpredado este concepto.

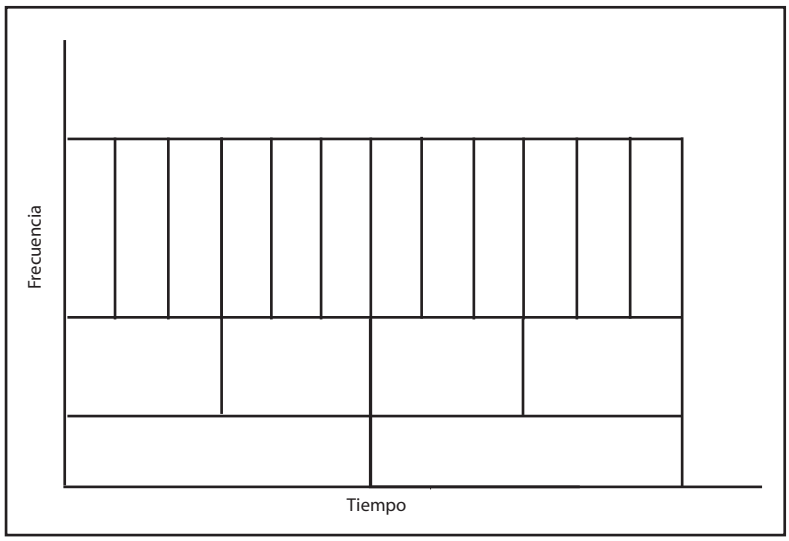


Figura 3.6: Esquema de la resolución del plano tiempo-frecuencia.

A partir de esta figura se nota que ninguna de las cajas posee un área igual a cero, lo cual implica que el valor de un punto en particular en el plano tiempo-frecuencia no puede ser conocido. Todos los puntos en el plano tiempo-frecuencia que están dentro de una caja son representados por un valor de la transformada wavelet.

Otro aspecto a notar en la figura es que a pesar de que las dimensiones de las cajas cambian, su área es constante, esto es que cada caja representa una porción igual del plano tiempo-frecuencia, pero en distintas porciones. En la figura se observa que para bajas frecuencias su altura es pequeña, lo que implica que poseen resolución en frecuencia pero el ancho es muy amplio, lo que implica que tienen poca resolución en el tiempo; para las altas frecuencias ocurre lo contrario, estas poseen pobre resolución en frecuencia, pero tienen buena resolución en el tiempo.

La transformada discreta wavelet

Aunque la discretización de la transformada wavelet continua posibilita el cálculo de la transformada wavelet continua, esta no es una verdadera transformada discreta, esta es solo una versión muestreada de la transformada continua, y la información que provee es altamente redundante en lo que a la reconstrucción de la señal se refiere. Esta redundancia al mismo tiempo aumenta el tiempo de cálculo y los recursos utilizados. La transformada discreta wavelet (DWT) por otro lado provee suficiente información tanto para el análisis como para la síntesis de la señal original, con una significativa reducción en el tiempo de cálculo.

La idea principal detrás del cálculo de la DWT es la misma que con la CWT. Una representación en el espacio tiempo-escala es obtenida utilizando técnicas de filtrado. La CWT se cálculo cambiando la escala de la ventana de análisis y desplazando la ventana en el tiempo, multiplicando la señal e integrando sobre todos los tiempos. En el caso discreto, filtros de diferentes frecuencias de corte son utilizadas para analizar la señal a diferentes escalas. La señal es pasada a través de una seria de filtros paso alto para analizar altas frecuencias y es pasada a través de filtros paso bajo para analizar las bajas frecuencias.

La resolución de la señal, que es una medida de la cantidad de detalle de la señal, es alterada por las operaciones de filtrado y la escala es cambiada por operaciones de sobre-muestreo y sub-muestreo. Sub-muestrear una señal corresponde a reducir la señal el rango de muestreo o remover algunas muestras de la señal. For ejemplo sub-muestrear por 2 significa eliminar la mitad de las muestras de la señal. Sub-muestrear por un factor n reduce el número de muestras de la señal por n veces. Sobre-muestrear la señal corresponde a incrementar el muestreo de la señal agregando nuevas muestras a la señal.

Para observar el proceso se tomará un ejemplo. Supóngase que se tiene una señal original $x[n]$ que tiene 512 muestras, abarcando una banda de frecuencias desde o a π rad/s. Al primer nivel de descomposición, la señal es pasada a través de filtros paso alto y paso bajo, seguidos por un sub-muestreo por 2; la mitad de las muestras pueden ser eliminadas de acuerdo a la regla de Nyquist, debido a que la señal cubre ahora solo la mitad de la frecuencia $\pi/2$, con esto la escala de la señal es doblada. Nótese que el filtro paso bajo remueve la información de las altas frecuencias, pero no cambia la escala, solo con el proceso de sub-muestreo la escala cambia. Sin embargo, el proceso de sub-muestreo no afecta la resolución debido que el remover la mitad de los componentes espectrales de la señal hace que la mitad de las muestras sean redundantes, la mitad de las muestras puede ser descartada sin perdida de información.

Continuando con el ejemplo, la salida del filtro paso alto tiene ahora 256 muestras (entonces la mitad de la resolución en el tiempo), pero solo cubre las frecuencias desde $\pi/2$

hasta π rad/s, entonces doblando la resolución en frecuencia. Estas 256 muestras constituyen los coeficientes de primer nivel de la DWT. La salida del filro paso bajo iene 256 muestras, pero también oslo cubre la mitad de las frecuencias,desde 0 hasta $\pi/2$ rad/s. La señal es pasada de nuevo a través de los filtros paso alto y paso bajo. La salida del filtro paso alto tendrá 128 muestras, con una frecuencia de $\pi/4$ a $\pi/2$ rad/s y la salida del filtro paso bajo tendrá también 128 muestras cubriendo frecuencias desde 0 hasta $\pi/4$. La señal tiene ahora la mitad de la resolución en el tiempo, pero el doble de la resolución en la frecuencia que en el primer nivel, esto es la resolución en el tiempo ha decrecido por un factor de 4 y la resolución en freccuencia ha aumentado igualmente por un factor de 4.

Para este ejemplo en específico se tendrán ocho niveles de descomposición, cada uno con la mitad de frecuencias del anterior, el proceso termina cuando se tiene solamente una muestra en la salida del filtro paso bajo. Nótese que al final del proceso se van a tener la misma cantidad de muestras que la señal original, esparcidas a través de todos los niveles de decomposición. En la figura 3.7 se muestra el procedimiento de descomposición de la señal $x[n]$ y en la figura 3.8 se muestra el ejemplo de una señal con 256 muestras y su DWT, nótese que esta representación muestra los distintos niveles de descomposición de la señal original.

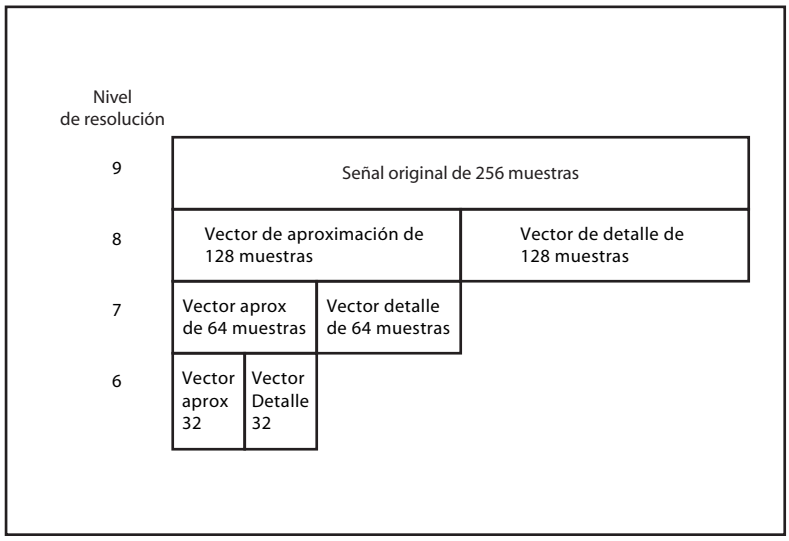


Figura 3.7: Procedimiento de descomposición de una señal en sus coeficientes wavelets.

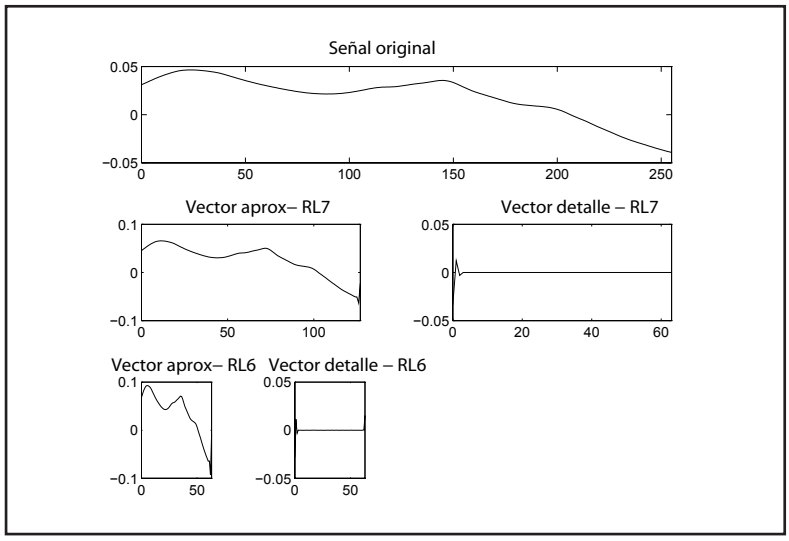


Figura 3.8: Ejemplo de aplicación de la DWT.

3.2.5 Los Modelos Ocultos de Markov

Los Modelos Ocultos de Markov son una clase de modelos para aproximar la densidad de probabilidad de una secuencia de símbolos observados. Estos son esencialmente máquinas de estado estocásticas que produce un símbolo cada vez que estos cambian de estado. Especificando las probabilidades de transición de estado entre los estados y el modelo de observación de símbolos para cada estado, es posible intentar capturar la estructura oculta en una larga cadena de símbolos. En general, el procedimiento consiste en seleccionar un estado de inicio acorde a alguna distribución de probabilidad. En cada paso, generar un símbolo de salida por medio de modelo generativo del estado actúal, y la transición a un nuevo estado de acuerdo a una matriz de probabilidad de transición estática.

Los Modelos Ocultos de Markov pueden ser discretos o continuos y pueden tener desde 1 hasta varios ordenes, claro esta los modelos continuos y de varios ordenes son mas difíciles de analizar y de aplicar. En este caso se hará mención a los modelos discretos de orden uno que son los mas simples y utilizados en aplicaciones.

Los Modelos cultos de Markov discretos

Sea una cadena de Markov de N posibles estados. El sistema puede ser descrito por estar en alguno de los N distintos estados, $s_1, s_2, \cdots, s_N$. La correspondiente HMM para observaciones de símbolos discretos está caracterizada por lo siguiente:

1. N, el número de estados del modelo. Estos están interconectados de tal forma que todos los estados puedan pasar a cualquier otro de los estados.

2. M, el número de distintos símbolos de observación por estado. Se denotan los símbolos individuales como $V = \{v_1, v_2, \cdots, v_M\}$.

3. La distribución de probabilidad de transición de estado $A = [a_{ij}]$ donde

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i), 1 \leq i, j \leq N \tag{3.4}$$

   donde q_t denota el estado actual en el tiempo t.

4. La probabilidad de distribución de observación de símbolos. $B = [b_{jm}]$, en la cual

$$b_{jm} = P(o_t = v_m | q_t = S_j), 1 \leq m \leq M \tag{3.5}$$

   define la distribución de símbolos en el estado s_j, y o_t es la observación en el tiempo t.

5. La distribución inicial de estado $\pi = [\pi_i]$ en la cual

$$\pi_i = P(q_i = S_i), 1 \leq i \leq N \tag{3.6}$$

Una especificación completa de un HMM requiere entonces de tres conjuntos de probabilidad, A, B y π. Para conveniencia se utiliza la siguiente notación

$$\lambda = (A, B, \pi) \tag{3.7}$$

En la figura 3.9 se muestra un modelo de primero orden con tres estados y la distribución de las matrices A, B y π.

Habiendo definido formalmente los HMMs, es posible dar un vistazo a los tres problemas clásicos que deben ser resueltos en orden de aplicar los modelos a problemas de la vida real, de estos tres problemas solamente el primero y el tercero son de interés para el desarrollo de este proyecto, por lo que solamente estos dos serán un poco más analizados.

Los tres problemas son:

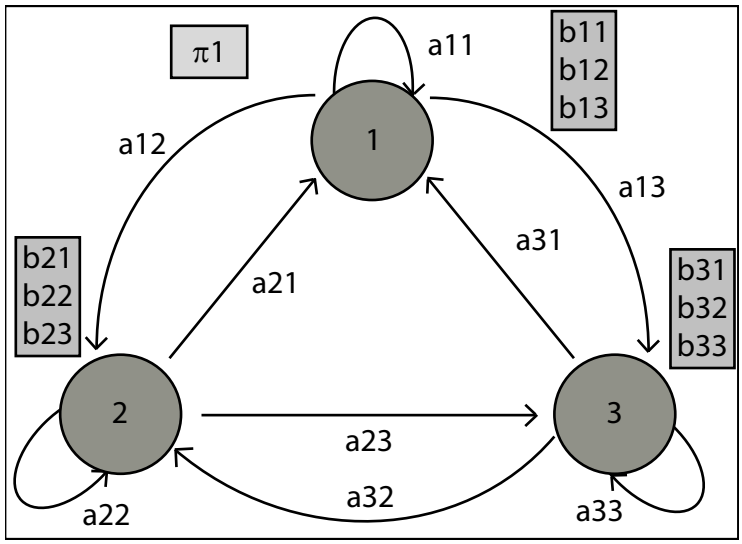


Figura 3.9: Ejemplo de un HMM con tres estados y sus matrices A, B y π.

1. Evaluación: Esto corresponde al cálculo de $P(S|\lambda)$, o sea la probabilidad de que el modelo sea capaz de generar una secuencia de observación. El principal detalle acá es la eficiencia computacional, encontrar un algoritmo computacional que requiera solamente un tiempo de cálculo polinomial.

2. Desciframiento: Este problema corresponde a encontrar la secuencia de estados ocultos que mejor corresponda a la secuencia de observación. Debido a que existen muchas secuencias que pueden generar la misma secuencia de observación, no existe una solución correcta que se pueda encontrar en muchos casos. Entonces se debe de encontrar el camino que maximiza $P(S|\lambda)$.

   Como se mencionó anteriormente este problema de los HMMs no es de interés para la solución del proyecto ya que para el reconocimiento de secuencias no es util la información del camino seguido para generar esa secuencia.

3. Entrenamiento: Esto corresponde a encontrar los parámetros del modelo $\lambda = (A, B, \pi)$ que especifica que el modelo con mas tendencia a producir una secuencia dada de datos de entrenamiento. No existe una forma analítica para poder encontrar el mejor modelo, sino que se aplican algoritmos iterativos para encontrar la mejor solución.

Para analizar el primer problema sea alguna secuencia de símbolos σ la c ual es τ pasos de tiempo de longitud:

$$\sigma : \sigma_1, \sigma_2, \cdots, \sigma_\tau \tag{3.8}$$

Existen muchas posibles trayectorias de estado que pueden producir esta misma secuencia, lo que se desea es encontrar todas estas trayectorias y obtener todas sus probabilidades. Sea el conjunto de trayectorias de estado que posiblemente producen a σ por medio de $I = \{i_1, i_2, \cdots, i_Q\}$. aquí cada elemento i de I es una trayectoria de estados que posee una longitud de τ estados que representa los estados que el sistema visitó (en orden) mientras se produjo la secuencia de símbolos σ. El estado visitado en el tiempo t en una trayectoria i es denotada por $i(t)$. Ahora se define la probabilidad de que un modelo en particular M causara la secuencia σ como

$$\begin{aligned} L^{\sigma}(M) &= Prob[\sigma_1, \sigma_2, \cdots, \sigma_\tau|\lambda] \\ &= \sum_{i \in I} T_{0i(1)} A_{i(1)}(\sigma_1) T_{i(1)i(2)} A_{i(2)}(\sigma_2) \cdots T_{i(\tau-1)i(\tau)} A_{i(\tau)}(\sigma_\tau) \end{aligned} \tag{3.9}$$

Esta es una expresión con un alto costo computacional, requiriendo $O(\tau N^\tau)$ multiplicaciones y $O(N^\tau)$ sumas, sin tomar en cuenta el trabajo que toma calcular cuales secuencias deben ser usadas en I. Sin embargo existe una salida a este problema.

Definase $\alpha_j(t)$ como las probabilidades incrementales de estar en el estado s_j en el tiempo t habiendo emitido la secuencia correcta de símbolos hasta ese momento (incluyendo σ_t)

$$(\alpha_t)^{\sigma}(j) = P(\sigma_1, \sigma_2, \cdots, \sigma_t|s = s_j|t = t_i) \tag{3.10}$$

Ahora la inducción produce que

$$(\alpha_1)^{\sigma}(j) = T_{0j} A_j(\sigma_1) \tag{3.11}$$

$$(\alpha_{t+1})^{\sigma}(k) = \sum_{s_j \in S} \alpha_t^{\sigma}(j) T_{jk} A_{k\sigma_{t+1}} \tag{3.12}$$

Lo que posibilita calcular fácilmente la probabilidad deseada $L^{\sigma}(M)$ dado que ahora se sabe que se debe finalizar en algún posible estado.

$$L^{\sigma(M)} = \sum_{s_k \in S} \alpha_t^{\sigma}(k) \tag{3.13}$$

Esta expresión puede ser calculada con solo $O(N)$ multiplicaciones y $O(N^2\tau)$ sumas y no es necesario calcular los miembros del conjunto I.

El problema del entrenamiento para los HMMs es el de estimar las probabilidades de transición, la distribución inicial de estado las distribuciones de probabilidad de emisión a partir de datos de entrenamiento. La idea es que el model se luego presentado con datos de una fuente desconocida, y este reconozca los datos si estos tienen características similares a las de los datos de entrenamiento. Típicamente se tienen múltiples modelos, todos entrenados para representar una fuente por separado. En la fase de reconocimiento, los modelos son presentados cada uno con los datos de evaluación para obtener cual es el que mejor corresponde con los datos. Los datos pueden ser entonces clasificados como pertenecientes a la fuente cuyo modelo sea el mas representativo de los datos.

Para el entrenamiento, varias soluciones matemáticas diferentes pueden ser utilizadas. La tarea es optimizar el modelo o un conjunto de modelos con respecto a algún criterio de optimización. Si el criterio de optimización es el de maximizar la probabilidad $P(O|\lambda)$, donde O son las secuencias de entrenamiento, una solución practica corresponde al algoritmo de Baum-Welch.

El algoritmo de Baum-Welch (presentado acá de una forma esquemática) es una procedimiento iterativo que progresivamente depura los parámetros (A, B, π) hasta que un máximo en la probabilidad P es alcanzado (nótese que dado que Baum-Welch es un algoritmo iterativo requiere de condiciones iniciales, que es otro aspecto a considerar en el diseño de la solución). El proceso es mostrado en la figura 3.10, nótese que los parámetros del bucle anterior son utilizados para el cálculo del siguiente. Por esto el procedimiento es frecuentemente llamado reestimación. Nótese además que la probabilidad del modelo es estrictamente creciente, en cada paso la probabilidad es mejorada o el proceso termina, lo que implica que es posible no encontrar máximos absolutos de la función.

Ejemplo de aplicación de los HMMs

Supóngase que se tienen dos personas jugando a los dados a través de un chat, una de las personas se encuentra en Santiago de Chile y la otra se encuentra en Londres, Inglaterra. La primera persona se encarga de hacer las tiradas y de informarle a la otra persona cuales fueron los resultados de las tiradas, de forma que esta solamente puede ver cual es la secuencia de resultados que se obtuvieron de las tiradas.

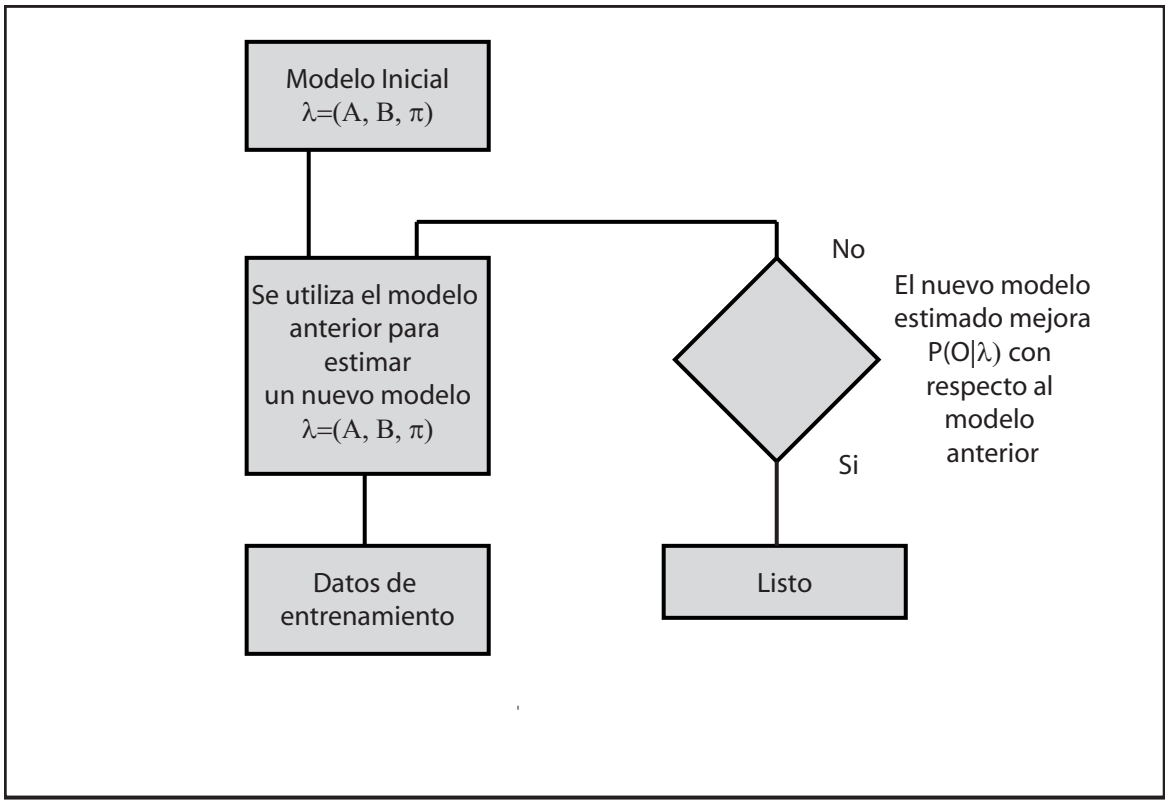


Figura 3.10: Esquema del algoritmo de Baum-Welch.

Supóngase ahora que la persona en Santiago utiliza dos dados, uno de los cuales tiene dos de sus lados cargados para hacer trampa, esta persona de vez en cuando de manera aleatoria y sin aviso substituye el dado bueno por el dado cargado para realizar las tiradas. De nuevo la otra persona solamente esta viendo las secuencias de tiradas que son resultado de las tiradas de un dado, pero cada secuencia puede ser el resultado de haber tirado el dado cargado o bien el dado bueno, sin saber en que momento se hizo el cambio de los dados. Entonces como puede saber la persona de Londres que la otra persona le esta haciendo trampa?

La respuesta a este problema es utilizar los HMMs para conocer cual fue el camino de tiradas que produjeron la secuencia que posee el jugador de Londres. Existe múltiples alternativas para plantear el modelo que puede ser utilizado. Una alternativa sencilla es tomar dos posibles estados bien definidos, los dos estados corresponden primero al dado bueno y segundo al dado cargado. La matriz de transición A muestra que es más probable permanecer en el estado 1 que pasar al estado 2 y quedarse ahí. La matriz de probabilidad de observación B, para el estado 1, muestra que las probabilidades para cualquiera de los dados es la misma, mientras para el caso del dado cargado se muestra claramente que los dos lados cargados tendrán una mayor tendencia a aparecer con respecto a los otros

lados. La matriz de estado inicial π muestra que cualquiera de los dos dados puede ser utilizado para la primera tirada.

3.2.6 Quantización vectorial

La Quantización Vectorial (VQ) es un método de compresión que no permite la reconstrucción exacta de los datos, esta basado en el principio de codificación por bloques. Esta es utilizada en muchas aplicaciones como es el caso de compresión de imágenes y de voz, reconocimiento de voz y en general en reconocimiento estadístico de patrones.

En el pasado el diseño de un quantizador vectorial era un problema de grandes dimensiones debido a la necesidad de resolver integrales multi-dimensionales. En 1980 Linde, Buzo y Gray propusieron un nuevo algoritmo de VQ basado en secuencias de entrenamiento. El uso de secuencias de entrenamiento evade el tener que acudir a las integrales multi-dimensionales.

Un quantizador vectorial no es mas que un aproximador. La idea es similar a la del redondeo a enteros, en donde se asigna el entero más cercano para un valor en particular que está en medio de los valores enteros que en este caso son los quantizadores.

Quantizadores vectoriales

El quantizador vectorial mapea vectores de dimensión k en el espacio vectorial R^k dentro de un conjunto finito de vectores $Y = y_i; i = 1, 2, \cdots, N$. Cada vector y_i es llamado un vector de código o "codeword". Al conjunto de todos los codewords se les llama "codebook". Asociado con cada codeword, y_i, se delimita una región cercana llamada región de Voronoi, la cual se define como

$$V_i = x \in R^k :\| x - y_i \| \leq \| x - y_j \|, \forall j \neq i \tag{3.14}$$

El conjunto de regiones de Voronoi parte el espacio R^k de modo que

$$\bigcup_{i=1}^{N} V_i = R^k \tag{3.15}$$

$$\bigcap_{i=1}^{N} V_i = \varnothing \tag{3.16}$$

Sin perder generalidad, en la figura se muestra algunos vectores en el espacio, asociado con cada grupo (cluster) de vectores se tiene un codeword representativo. Cada codeword reside en su propia región de Voronoi. Estas regiones están separadas por líneas imaginarias en la figura.

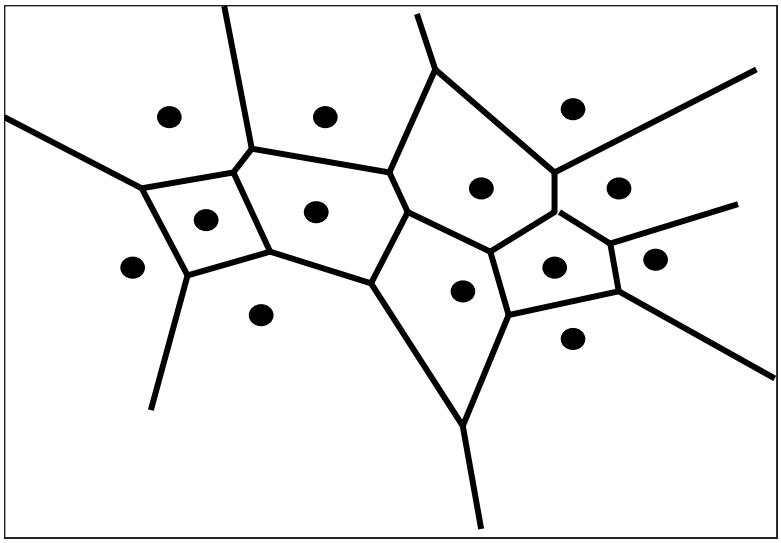


Figura 3.11: Ejemplo de un conjunto de vectores en dos dimensiones y sus regiones de Voronoi.

El algoritmo LBG

Dado un conjunto de vectores de entrenamiento con sus propiedades estadísticas conocidas, se describe una medida de distorción, que corresponde a la distancia que máxima que existe entre el codeword y los vectores en la región de Voronoi, así como el número de miembros del codebook a crear. Se busca encontrar encontrar el codebook y la partición de regiones de Voronoi con la menor distorción promedio.

Se asume entonces que se tienen M vectores de entrenamiento $X = x_1, x_2, \cdots, x_M$. M se debe de tomar suficientemente largo para que todas las propiedades estadísticas de la fuente sean capturadas por las secuencias de entrenamiento. Se asume que los vectores pertenecen al espacio R^k para algún k conocido. Sea N el número de miembros del codebook, donde $C = c_1, c_1, \cdots, c_M$ representa el codebook. Cada codeword se asume que también pertenece al espacio R^k. Sea además las N regiones de Voronoi asociadas

con los miembros del codebook, dadas por $P = S_1, S_2, \cdots, S_N$. Si el vector x_m esta en la región S_n, entonces su aproximación o quantizador denotado por $Q(x_m)$ es c_n

$$Q(x_m) = c_n, x_m \in S_n \tag{3.17}$$

El algoritmo LBG tiene los siguientes pasos:

1. Escoger ϵ y N
2. Sea $N = 1$. El quantizador inicial corresponde al centro de masa de toda el área, con esto se obtiene el punto de referencia inicial

$$(\tilde{x}_1)^* = \frac{1}{M}\sum_{t=1}^{M} x_t \tag{3.18}$$

   y la distorsión promedio

$$D^*_{avg} = \frac{1}{Mk}\sum_{t=1}^{M}(\parallel x_t - \tilde{x}^*_1 \parallel)^2 \tag{3.19}$$

3. Se separa cada quantizador en dos, de forma que divide el espacio vectorial en mitades conforme va avanzando el algoritmo. Entonces para cada $n = 1 \cdots N$

$$\tilde{x}^{(0)}_n = (1 + \epsilon)\tilde{x}^*_n \tag{3.20}$$

$$\tilde{x}^{(0)}_n = (1 - \epsilon)\tilde{x}^*_n \tag{3.21}$$

   Se actualiza $N = 2N$

4. Se itera para optimizar los quantizadores, se inicializa

$$D_{avg}^{0} = D_{avg}^{*} \tag{3.22}$$

$$i = 0 \tag{3.23}$$

4.1. Se encuentra para cada punto de entrenamiento el quantizador mas cercano, para cada $m = 1 \cdots M$ se encuentra el valor minimo de

$$(\parallel x_m - c_n^{(i)} \parallel)^2 \tag{3.24}$$

para todos los $n = 1, 2, \cdots, N$. Sea n^* ser el index que produce este valor mínimo, entonces

$$Q(x_m) = c_{n^*}^{(i)}, \tag{3.25}$$

4.2. Para $n = 1, 2, \cdots, N$ se actualiza el codeword, esto obtimiza la ubicación del codeword

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m)=c_{n^*}^{(i)}} x_m}{\sum_{Q(x_m)=c_{n^*}^{(i)}} 1}, \tag{3.26}$$

4.3. Se toma $i = i + 1$

4.4. Se calcula

$$D_{ave}^{*} = \frac{1}{Mk} \sum_{m=1}^{M} (\parallel x_m - Q(x_m) \parallel)^2 \tag{3.27}$$

4.5. si $(D_{ave}^{i-1} - D_{ave}^{i})/D_{ave}^{i-1} > \epsilon$, se pasa al punto 1.

4.6. se toma $D_{ave}^{*} = D_{ave}^{i}$. Para $n = 1, 2, \cdots, N$ se hace

$$c_n^{*} = c_n^{(i)} \tag{3.28}$$

como el codebook final

5. Se repiten los pasos 3 y 4 hasta alcanzar el número deseado de codewords.

3.2.7 Mapero no lineal de Sammon (Sammon's Mapping)

El mapeo no lineal de Sammon intenta encontrar una representación de baja dimensión (normalmente dos o tres dimensiones) de un conjunto de puntos distribuidos en un espacio de patrones de alta dimensión, de modo que las distancias euclideanas entre los puntos en el mapa es similar a la posible distancia entre los puntos correspondientes en el espacio de patrones de alta dimensión.

En la figura 3.12 se muestra el mapeo de Sammon en dos dimensiones para un conjunto de datos de la distribución de extractos de polen de la flor de Liz, los datos son de 4 dimensiones, el cual muestra una buena visión de la distribución de los datos.

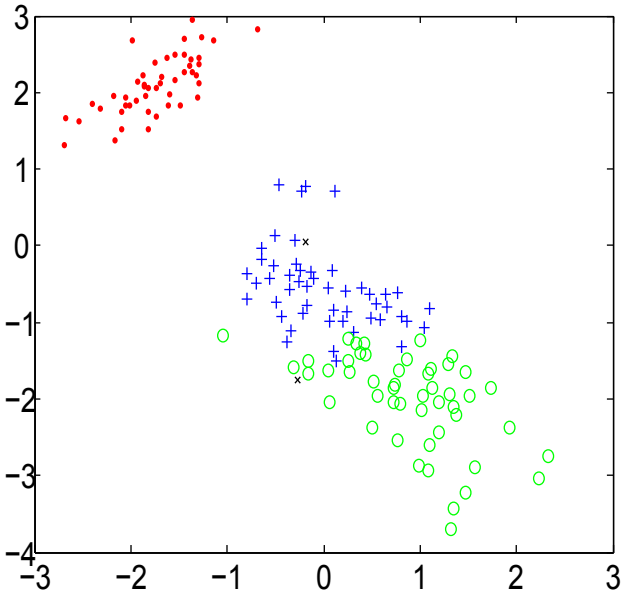


Figura 3.12: Mapeo de Sammon para un conjunto de datos de la distribución de extractos de polen de la flor de Liz.

Capítulo 4

Procedimiento Metodológico

4.1 Reconocimiento y definición del problema

Para el reconocimiento del problema se realizaron las siguientes actividades las cuales conformaron un lapso de aproximadamente 1.5 meses:

1. Estudio de los principales principios relacionados con los sistemas de refrigeración y los principios de transferencia de calor, con el objetivo de familiarizarse con el entorno en el cual de desarrolla el proyecto y obtener las principios básicos que rigen el problema a resolver.

2. Visitas periódicas a los laboratorios de estrés de Q&R, en donde se obtuvieron los principios de funcionamiento de las cámaras, sus principales componentes y las funciones que desempeña, así como tener contacto con los distintos tipos de cámaras de choque térmico, esto a cargo de los asesores y encargados del laboratorio.

3. Estudio de los diagramas eléctrico, mecánico y de válvulas de las cámaras con el fin de visualizar y entender el funcionamiento de los distintos componentes que conforman la cámara, para obtener una justificación de su uso a partir de la teoría de los sistemas de refrigeración.

4. Entrevista con los encargados de mantenimiento de las cámaras de la empresa proveedora, a partir de la cual se obtuvieron los principales problemas que presentan las cámaras.

5. Encuesta con los encargados de la administración de los laboratorios con el fin de obtener una idea global del impacto que tendrá la resolución del problema y a partir de esto definir una meta clara para los resultados del proyecto.

6. Sesión de asesoría con los proveedores de los componentes a utilizar en Costa Rica, para la aclaración de dudas y análisis de las soluciones planteadas.

4.2 Evaluación de las alternativas y síntesis de una solución

Para realizar el planteamiento de la solución se siguieron las siguientes actividades:

1. Estudio de las tecnologías envueltas en el funcionamiento de las cámaras, obteniendo el conocimiento y criterio necesario para el planteo de las soluciones y la selección de los componentes requeridos.
2. Investigar los principales proveedores de componentes de forma que se adecuaran a las necesidades de la empresa en cuanto a precio y confiabilidad, y también a las necesidades de tiempo impuestas para el proyecto.
3. Realizar una investigación de los recursos con los que ya cuenta la empresa para la realización del proyecto.
4. Estudio de los principios matemáticos que están integrados en la herramienta de predicción de fallas ya implementada, de forma que se pudiera entender su estructura y la metodología requerida para implementarla.

Para la evaluación de la solución se realizarán encuentros, en donde se harán presentes los encargados de la administración del laboratorios y los encargados de mantenimiento del mismo, en donde será expuesta la solución de forma total y se obtendrán las principales recomendaciones por parte de los asistentes. Además de esto es necesaria la coordinación requerida en cuanto a horarios de trabajo, recursos y planta física necesaria para la realización del proyecto, además dejar establecidas las pautas y procedimientos necesarios para la realización del proyecto, tanto en la etapa de experimentación como en las de implementación del mismo.

4.3 Implementación de la solución

Para la implementación de la solución primero será requerida la convocatoria periódica de reuniones con el fin de exponer los procedimientos que se realizarán previos a su ejecución con el fin de coordinar las actividades y garantizar el éxito de las mismas.

Las actividades que serán llevadas a cabo para evaluar los módulos a implementar son las siguientes:

1. Previamente a su realización serán realizadas pruebas de laboratorio y de campo de forma que se asegure el funcionamiento de cada modulo desarrollado antes de ser implementados en las cámaras, así mismo se deberá obtener la autorización adecuada de las autoridades respectivas de la empresa.

2. En el caso de los módulos de software serán expuestos a pruebas de rendimiento (vectores de prueba) que permitan la detección de fallas de programación (pulgas) o cualquier otro problema que no se haya encontrado.

3. Para los sensores adquiridos serán realizadas pruebas de laboratorio en donde se puede comprobar su funcionamiento de acuerdo al que el fabricante garantiza.

4. Estudio de los principios matemáticos requeridos para el desarrollo de la herramienta de condicion. La evaluación de los modelos de predicción será cotejada a partir de simulaciones y pruebas de laboratorio con equipos que permitan verificar la eficiencia de los modelos y del entrenamiento realizado.

Capítulo 5

Descripción detallada de la solución

En este capítulo se describe en detalle la solución al problema de la integración de las cámaras de choque térmico al sistema LabMonitor. En este se explica el trabajo realizado en dos tareas de distintas áreas pero que tienen como objetivo el poder llevar información útil que permita emitir criterios acerca de la condición de las cámaras y poder tomar acciones correctivas si es del caso. El trabajo realizado hace uso de numerosas librerías y herramientas producidas como código libre (open source), por lo que se pretende no solamente demostrar el trabajo realizado durante el proyecto sino mostrar el trabajo que otros profesionales en el mundo están desarrollando y que resulta de gran utilidad en el desarrollo de nuevas tecnologías.

5.1 Análisis de las soluciones y selección final

La integración de las cámaras de choque térmico en el sistema LabMonitor requiere el envío de los reportes de fallas producidos por las cámaras a la base de datos del sistema, junto con esto es necesario el desarrollo de una herramienta que sea capaz de reconocer la condición de los compresores que estas poseen siendo estos elementos sensibles en el funcionamiento de las cámaras y que requieren de una atención minuciosa.

Esto requiere que la solución planteada tenga dos partes. La primera parte requiere brindar comunicación a las cámaras con el sistema LabMonitor con el fin de obtener la información necesaria que genere los reportes a enviar a la base de datos de LabMonitor (para esto será necesario determinar que información es de importancia para la generación de reportes), y desarrollar la herramienta que permita a LabMonitor generar los reportes a partir de la información obtenida.

La segunda parte es el desarrollo de una herramienta para evaluar la condición de los

compresores de las cámaras de choque térmico. Esta herramienta se realizó en base a investigaciones anteriores y recomendaciones hechas en proyectos anteriores que no fueron terminados, los cuales no se mencionarán acá. La herramienta se basa en la investigaciones realizadas para el desarrollo de software para el monitoreo de equipos industriales, principalmente en el trabajo titulado "Hidden Markov Model-based Tool Wear Monitoring In Turning" a cargo de Litao Wang, Mostafa G. Mehradi y Elijah Kannatey-Asibu, Jr., este trabajo se detalla en [18].

5.1.1 Recolección de los reportes de fallas de las cámaras de choque térmico

Las cámaras de choque térmico cuentan con un conjunto de sensores que permiten medir variables tales como temperatura, nivel de humedad, presión, nivel de aceite de los compresores y otras. A partir de estas variables se generan un reportes de fallas que son gran importancia para controlar el funcionamiento y mantenimiento de las cámaras. Los reportes de los sensores son procesados y los resultados controlan el valor de posiciones de memoria en el PLC de las cámaras que son evaluados para generar los reportes,

Para visualizar los reportes, las cámaras cuentan con un panel táctil PanelView de Allen-Bradley. El panel se comunica con el PLC por medio de un canal de comunicación DH-485 (el cual es un protocolo de comunicación propietario de Allen-Bradley). Para determinar que mensaje debe ser agregado al reporte el panel realiza una lectura de las posiciones de memoria en el PLC que almacenan los resultados del procesamiento de variables. Estas posiciones de memoria debieron ser identificadas con el fin de reconocer que partes de la memoria deben ser leídas por parte de LabMonitor para generer los mismos reportes del panel de la misma forma en que este lo hace.

Como se mencionó en el párrafo anterior lo que se busco con la solución es que LabMonitor emulara el proceso que realiza el panel táctil para generar sus reportes de alarmas. Esto es realizar consultas de la memoria del PLC evaluar los resultados y generar los reportes, para luego enviarlos a la base de datos de LabMonitor.

Para brindar comunicación al PLC de las cámaras se optó por utilizar la red Ethernet de Intel, debido a que esta no requeriría el tendido de nuevos cableados y se comunica directamente con los servidores de LabMonitor. El modelo del PLC que se posee en las cámaras no posee medios para comunicarse a través de una red Ethernet, para esto las investigaciones con los fabricantes del PLC mostraron que existe una interfaz que se agrega al PLC y que permite utilizar un canal DF1 (este protocolo de comunicación es también propietario de Allen-Bradley) disponible y que realiza la conversión entre los dos tipos de redes, posibilitando la comunicación.

La interfaz utilizada es el 1761 NET-ENI de Allen-Bradley que como se mencionó es una interfaz que comunica redes DF1 con redes Ethernet. La comunicación con la interfaz se hace por medio del protocolo EthernetIP, el cual se describe en la sección 3.2.2.

Una vez lograda la comunicación con las cámaras, fue necesario desarrollar una herramienta que permitiera a LabMonitor leer la memoria del PLC. La comunicación entre LabMonitor y el PLC obedece a la interacción servidor-cliente a través del protocolo de aplicación CIP. Como se detalla en la sección 3.2.2, EthernetIP tiene como capa de aplicación al protocolo CIP de forma que fue necesario realizar una pequeña implementación del protocolo con el fin de que el PLC se comunicara con LabMonitor.

Para obtener las posiciones de memoria, así como los mensajes de texto fue necesario descargar y estudiar la aplicación del panel táctil la cual posee esa información.

La herramienta desarrollada para LabMonitor consiste de:

1. Una librería de uso de sockets TCP para C++.
2. La implementación de la capa de aplicación de EthernetIP la cual utiliza el protocolo CIP en conjunto utilizado para generar comandos para la lectura de la memoria.
3. Desarrollo de una aplicación en Java para la revisión de los resultados de las lecturas de memoria y generar los reportes de alarmas y almacenarlos en la base de datos MySQL de LabMonitor.
4. Uso de la herramienta Swig para poder comunicar las aplicaciones de C++ y Java, con el fin de usar la aplicación en C++ como una librería utilizada por la aplicación de revisión de Java.

En la figura 5.1 se muestra un esquema de la solución implementada. A manera de repaso general LabMonitor se comunica con el PLC a través de la interfaz 1761 NET-ENI por medio de la red Ethernet de Intel, utilizando la herramienta desarrollada que implementa la capa de aplicación del protocolo EthernetIP.

5.1.2 Desarrollo de la herramienta de reconocimiento de condición para los compresores de las cámaras de choque térmico

El propósito principal de la herramienta es dada una señal de entrada poder reconocer que tipo de fuente fue la que produjo esta señal, esto permite poder clasificar si la señal proviene de un compresor en buen estado o en mal estado y si se requiere poder reconocer

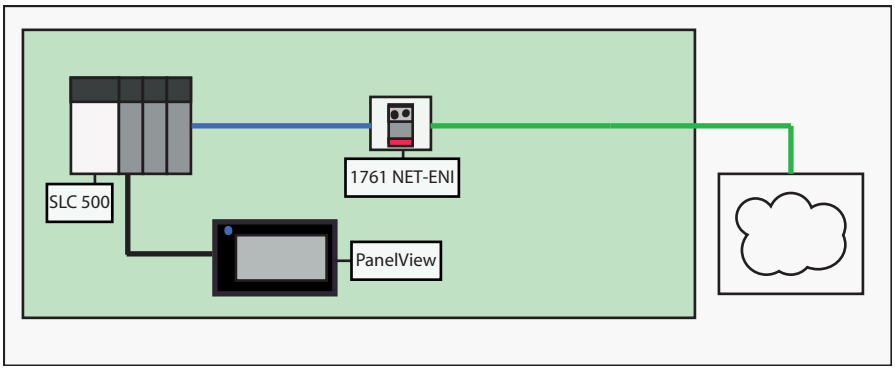


Figura 5.1: Esquema de la herramienta de reportes de fallas de las cámaras de choque térmico para LabMonitor.

si el problema que presenta el compresor es debido a un aumento de presión, falta de aceite, desgaste u otro problema más específico. El desarrollo de la herramienta debe permitir que los usuarios puedan aumentar el nivel de "conocimiento" que esta posee a través del entrenamiento continuo, esto significa que en el momento en el que se detecta un nuevo tipo de problema este pueda ser "aprendido" por la herramienta para ser reconocido en el futuro.

En la figura 5.2.2 y en las figuras 5.4 y 5.3 se muestra los esquemas que describe el proceso llevado a cabo por la herramienta de reconocimiento de condición tanto en el entrenamiento como en su evaluación. La herramienta hace uso de las señales de vibración que son emitidas por los compresores para poder reconocer su estado, estas señales son de las más características de los equipos industriales y se han utilizado en gran cantidad de herramientas de reconocimiento de condición.

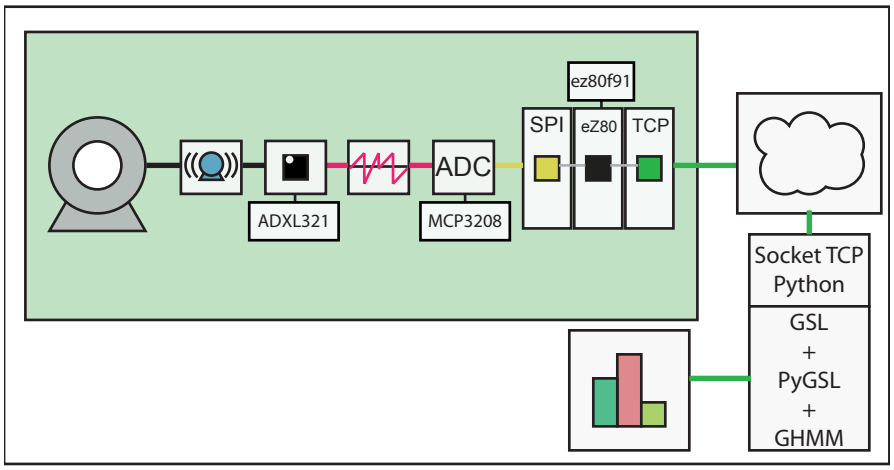


Figura 5.2: Diagrama de bloques de la herramienta de clasificación de condición.

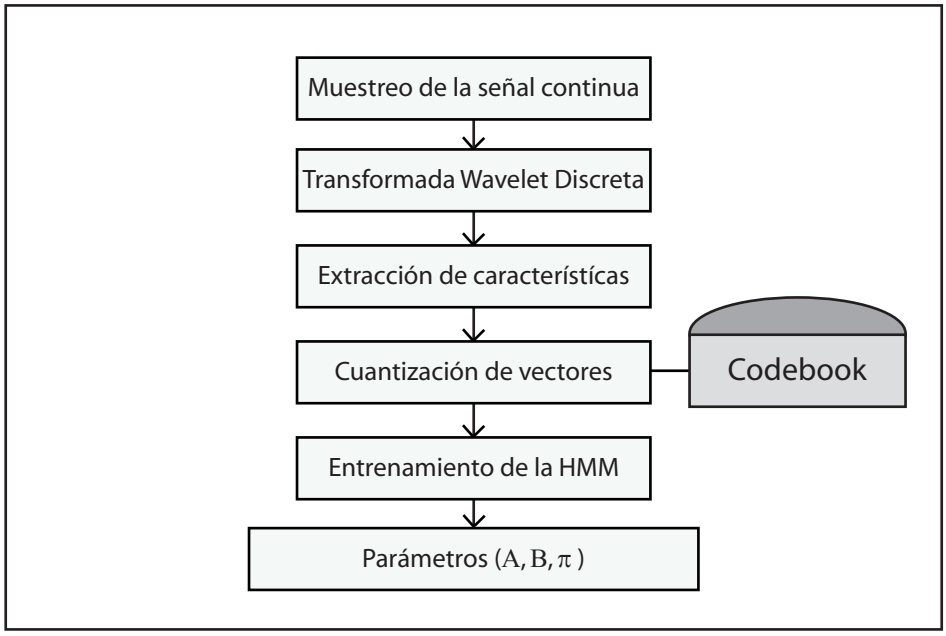


Figura 5.3: Esquema del proceso de entrenamiento de la herramienta de clasificación de condición.

En el primer paso del proceso se da el muestreo de la señal de vibración, por medio del hardware de adquisición de datos se muestrea una cierta cantidad de paquetes de muestras, el numero de muestras debe de ser un múltiplo de dos que permita tener suficientes niveles de frecuencia disponibles como se verá mas adelante, por lo que debe de tomarse en cuenta este aspecto a la hora de escoger la cantidad de muestras por paquete y el número de paquetes que corresponderán a la señal a evaluar. La cantidad de muestras varía de acuerdo a si las muestras serán utilizadas para entrenamiento o evaluación, en el primer caso se necesita una gran cantidad de muestras para poder recopilar suficiente variabilidad de la condición. Los pasos siguientes tendrán el propósito de convertir la señal muestreada en una secuencia discreta que será evaluada por los modelos de Markov.

Cada paquete de muestras obtenido corresponde a un miembro de la secuencia de entrenamiento o de evaluación, como se muestra en la figura 5.4, o viéndolo de otra forma, la señal a evaluar se divide en partes iguales, siempre con una cantidad de muestras igual a una potencia de dos, de forma que cada una de las partes en las cuales se divida la señal corresponde a un miembro de la secuencia, en el ejemplo mostrado para la señal A la longitud de la secuencia es 3.

En el siguiente paso cada miembro de la secuencia pasa de ser un grupo de muestras en el tiempo a ser cada uno un vector cuyos miembros serán las energías normalizadas de las bandas de frecuencias que componen la señal, para esto se hace uso de la DWT de

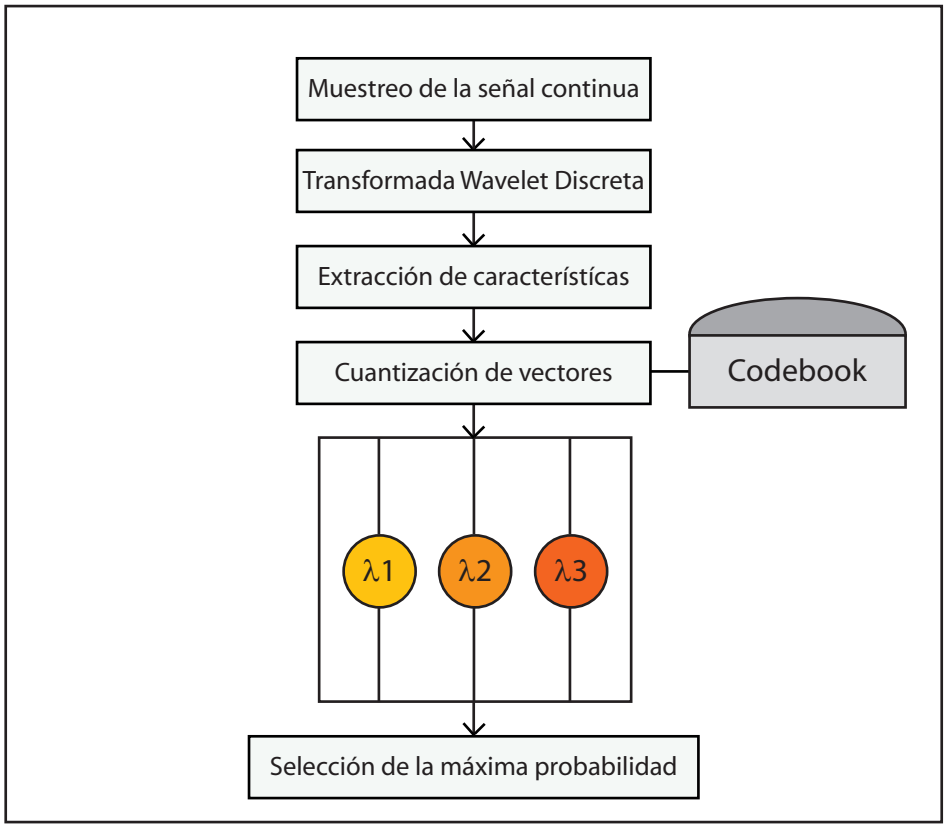


Figura 5.4: Esquema del proceso de la herramienta de clasificación de condición.

la señal de forma que está se divide en bandas de frecuencias, el número de bandas de frecuencias es igual a $log_2(n)$, donde n es el número de muestras que conforman el paquete a transformar. Hasta este punto el proceso es el mismo tanto para el entrenamiento como para la evaluación de la herramienta.

Algunas de las frecuencias pueden no ser tomados en cuenta debido a que posee muy pocas muestras debido al sub-muestreo. A continuación se calcula la energía de la banda de frecuencia, la cual se define por medio de

$$E_j = \frac{1}{T_j} \sum_{k=1}^{T_j} d_j(k)^2 \tag{5.1}$$

, donde $d_j(k)$ son los coeficientes wavelet de la señal para $j = 1, 2, \cdots, J$ y k representa el tiempo discreto; T_j es el número de coeficientes de cada escala. Las energía calculadas son

normalizadas para que los valores sean independientes de la señal. Con esto se tienen los miembros de la secuencia convertidos en vectores cuya dimensión es igual al número de escalas seleccionadas. Al producto obtenido hasta este momento se les llama vectores de características, y son aquellos vectores que muestran las características que hacen diferente a la señal.

Luego, el proceso se divide en dos dependiendo si se trata del entrenamiento o bien de la evaluación. Para el primer caso se hace uso del algoritmo LBG para calcular un codebook que agrupe todos los grupos de entrenamiento que se tienen disponibles, de forma que el codebook sea capaz de poder manejar todos los grupos disponibles. Los miembros del codebook corresponderán a los miembros del alfabeto de los modelos de Markov. Para obtener el tamaño del codebook es necesario realizar un análisis visual de la distribución de las muestras de entrenamiento, esto por medio del mapeo de Sammon en dos dimensiones de las secuencias de entrenamiento, de este modo se aproxima un valor para el tamaño del codebook.

En el caso de la evaluación, dado un conjunto de secuencias a evaluar, cada secuencia es entonces aplicada con el codebook y se obtiene entonces los índices correspondientes a los miembros del codebook que mapean la secuencia dentro del subconjunto de vectores que posee el codebook. Para esto se obtiene la distancia euclideana entre la secuencia de evaluación y cada uno de los miembros del codebook, aquel que produzca la menor de las distancias será aquel que describa mejor la secuencia, de forma que se obtiene una secuencia discreta, cual esta dentro del alfabeto de los modelos de Markov y haciendo la secuencia aplicable a los modelos.

Una vez obtenidas las secuencias, el siguiente paso es ya sea entrenar los modelos o bien aplicar los modelos existentes en la herramienta a la secuencia y obtener los valores de probabilidad generados por cada uno de los modelos, estos valores de probabilidad dan una idea de cual de los modelos pudo haber generado la secuencia y así reconocer cual es el estado de condición al que pertenece la secuencia de evaluación. En el caso del entrenamiento, es necesario tener muestras agrupadas de cada uno de los estados que se quieren reconocer y aplicarles el algoritmo de discretización hasta obtener una secuencia que los modelos puedan entender. Por medio de la librería GHMM que utiliza el algoritmo de entrenamiento de Baum-Welch se obtienen los parámetros del modelo $\lambda = (A, B, \pi)$. Para la evaluación una vez obtenidos los modelos, solo queda aplicar el reconocimiento y obtener los valores de probabilidad de cada modelo, dada una secuencia desconocida, la cantidad de valores de probabilidad obtenidos corresponderá a tantos modelos como se tenga, el mayor de estos valores establecerá cual es la fuente a la cual pertenece la secuencia desconocida.

5.2 Descripción del hardware

5.2.1 Comunicación del PLC a través de una red Ethernet

De acuerdo a la figura 5.1 el principal componente utilizado para integrar el PLC de las cámaras de choque térmico es la interfaz 1761 NET-ENI el cual conecta dispositivos con capacidades de comunicación DF1 en modo full-duplex a redes Ethernet. En la figura 5.5 se muestra un esquema de la interfaz y sus principales componentes. La interfaz brinda conexiones Ethernet a 10 y 100 Mpbs.

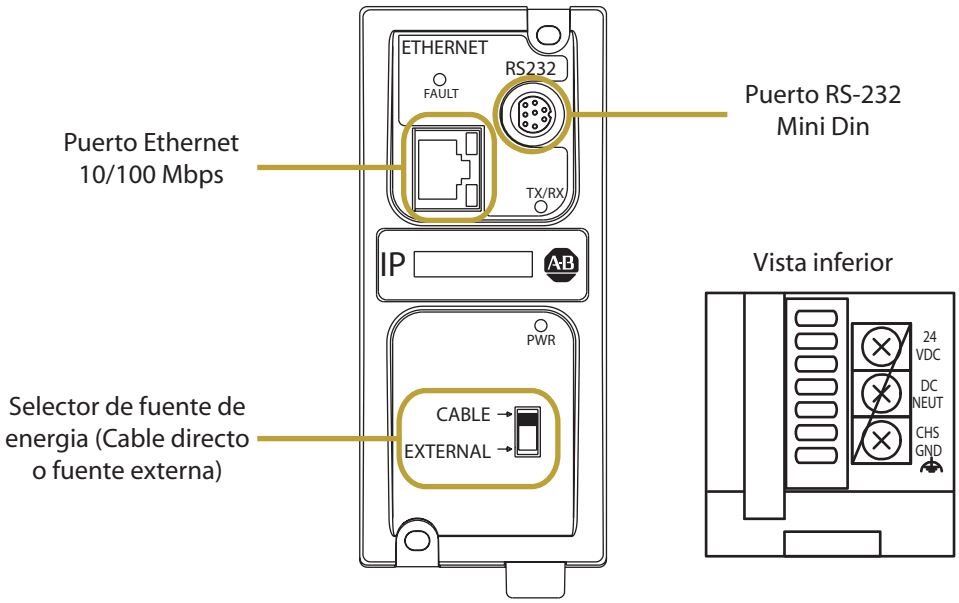


Figura 5.5: Esquema de la interfaz DF1-EthernetIP 1761 NET-ENI.

La implementación de la interfaz en el PLC es bastante directa. La alimentación es obtenida a partir de la fuente de alimentación del PLC la cual cuenta con una salida de 24 VCD y soporta una carga de hasta 3 A. La conexión con el canal DF1 del PLC se realizó por medio de un cable 1761-CBL-AP00, el cual convierte la conexión mini-din de 8 pines del NET-ENI en una conexión DB-9 estándar en el PLC.

5.2.2 Hardware de adquisición de datos

En la figura se muestra el diagrama de bloques del hardware utilizado en el sistema de adquisición de datos para las señales de vibración de los compresores de las cámaras de choque térmico

Como sensor de vibración se utilizó el acelerómetro ADXL321 de Analog Devices, el cual se muestra en la figura 5.6. En la figura 5.7 se muestra la tarjeta de evaluación ADXL321EB, el cual tiene montado el ADXL321 listo para ser utilizado. El ADXL321 es un acelerómetro de doble eje (eje x y eje y), de baja potencia ideal para aplicaciones de monitoreo, esto por su gran estabilidad en condiciones de cero vibración (cero g) y buena sensibilidad a los cambios. El ancho de banda del sensor se controla en un rango de 0.5 Hz hasta los 2.5 kHz, ajustables por medio de capacitores agregados en el montaje final. Además provee salidas de voltaje condicionadas que permiten una señal libre de ruido sin necesidad de dispositivos adicionales.

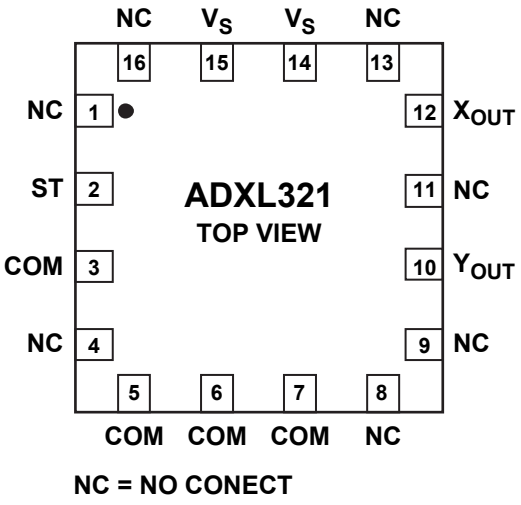


Figura 5.6: Esquema del acelerómtero ADXL321.

La tarjeta de evaluación posee capacitores de 100nF instalados de fábrica, lo que provee un ancho de banda de 50Hz. Debido a esto fue necesario substituir uno de los capacitores (específicamente el correspondiente al eje x, que fue el eje utilizado durante las mediciones) con un capacitor de 2200pF, el cual permitiera obtener el máximo ancho de banda permisible por el acelerómetro, 2500 Hz.

Para la escogencia del acelerómetro se tomo en cuenta el ancho de banda aproximado que se esperaba podrían presentar las señales de vibración de los compresores, así como el amplio rango de medición de aceleración que presenta el dispositivo. Finalmente su pequeño tamaño lo hace adecuado para la aplicación ya que permite una fácil instalación sin modificar demasiado el sistema.

Para muestrear la señal de voltaje obtenida del acelerómetro, se utilizó un convertidor analógico digital de Microchip Corporation MCP3208, mostrado en la figura 5.8. Este convertidor tiene una resolución de hasta doce bits, la cual es bastante adecuada para la tasa de cambio de la señal de vibración. Posee ocho canales de muestreo multiplexados,

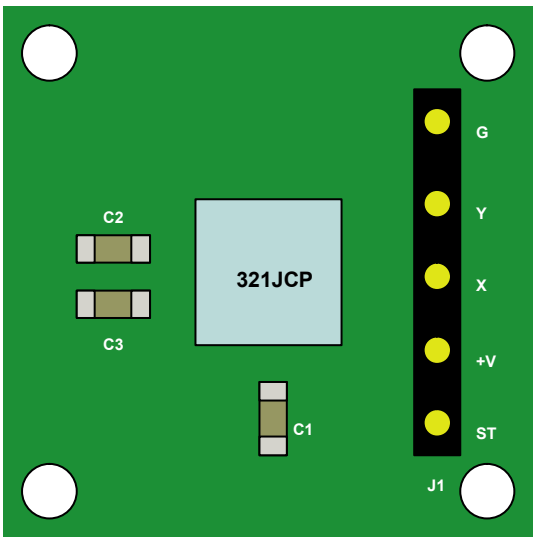


Figura 5.7: Esquema de la tarjeta de evaluación ADXL321EB.

de forma que es posible muestrear varios sensores con el mismo dispositivo. Posee una entrada para el voltaje de referencia, lo que permite adecuarse a la aplicación que se esta diseñando. Finalmente el dispositivo utiliza el protocolo SPI para la transmisión de datos

Para poder hacer llegar los datos sal servidor para el procesamiento se requiere de una puerta de salida para los datos a través de una red Ethernet. Para esto se utilizó el modulo ez80f91 de Zilog Corporation. En la figura 5.9, se muestra el diagrama de bloques y una imagen del módulo ez80f91 . Este mini modulo posee un microcontrolador ez80 de 50 MHz, es programable en lenguaje C, posee puerto Ethernet con capacidades de 10/100 Mbps full y half duplex, 4 puertos de propósito general, uno de ellos configurable para ser utilizado con el protocolo SPI.

De acuerdo a las especificaciones electricas del módulo ez80f91, el voltaje de salida que ofrecen los pines de alimentación (V_{DD}) es de aproximadamente 3.6 VCD, este voltaje resultó ideal para la alimentación tanto del acelerómetro como del convertidor analógico digital los cuales utilizan voltajes de alimentación tipicos cercanos a 3.5 VCD, esto permite, por un lado para el caso del acelerómetro voltajes de salida de aproximadamente 1.5 VCD para la condición de cero g y con una variabilidad de la señal entre 0.3 VCD y 2.6 VCD, por otro lado el voltaje de referencia aplicado al convertidor analógico digital proporciona un rango de señal de entrada de entre $V_{SS} = 0$ VCD y $V_{ref} = 3.7$ VCD, ambos aspectos provocan que la señal de salida del acelerómetro pueda ser muestreada en todo su rango de amplitud por parte del convertidor analógico digital. Para la comunicación entre el convertidor analógico digital y la puerta de salida se utiliza el puerto B del modulo de zilog configurado en su función alternativa, como se detallará más adelante se utiliza PB5 como chip select, PB7 como data output para el maestro, PB6 como data

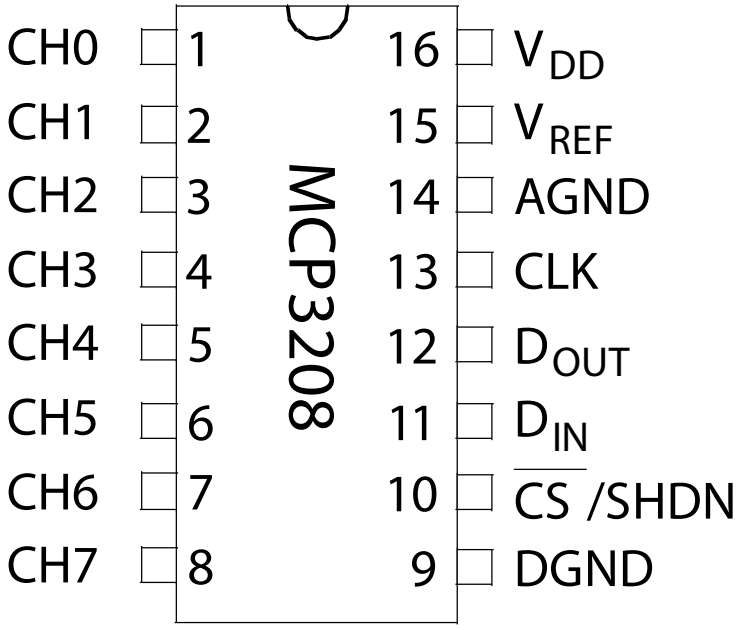


Figura 5.8: Convertidor analógico digital MCP3208.

input del maestro y PB3 como reloj.

5.3 Descripción del software

5.3.1 Libreria EthernetIP de C++

Como se mencionó anteriormente, la comunicación con la interfaz 1761 NET-ENI se realiza por medio del protocolo de aplicación CIP. De acuerdo a la documentación de Allen-Bradley y a los resultados obtenidos durante la experimentación, para el 1761-NET-ENI el puerto para mensajeria y transmisión de datos es el 44818, este es un puerto propietario de Allen-Bradley el cual también utilizan sus aplicaciones comerciales.

Para poder enviar comandos de lectura al PLC, estos deben de estar encapsulados dentro de un mensaje CIP, de esta forma el 1761 NET-ENI, al desencapsular el paquete reconoce cual es aquella porción del mensaje que debe de ser enviada al PLC a través del puerto DF1.

El proceso de conexión con la interfaz 1761 NET-ENI inicia con la solicitud de sesión

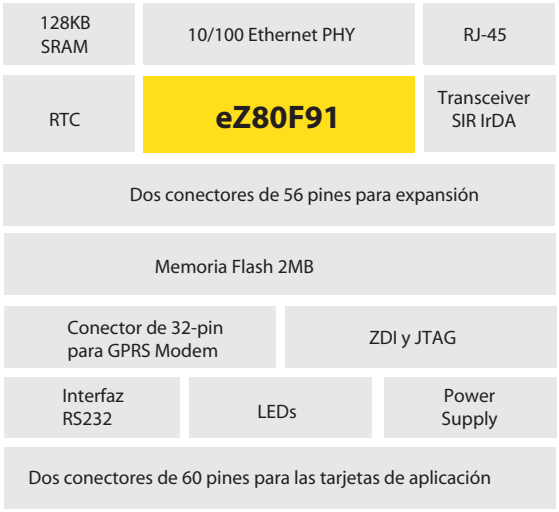


Figura 5.9: Kit de desarrollo ez80f91.

por parte del protocolo TCP, el cual conecta el cliente con la aplicación del servidor en el puerto 44818. Para iniciar la transmisión de datos en el protocolo CIP, es necesario la ejecución de tres comandos básicos, los cuales son detallados en la sección 3.2.2: listar servicios, listar interfaces y registrar la sesión, donde este ultimo devuelve el número de sesión que se utiliza para los comandos de solicitud de datos.

La implementación del cliente EthernetIP se desarrollo en C++ como una librería utilizable para software hecho en Java, esto debido a la facilidad que presenta C++ para el manejo de datos de bajo nivel y a la rapidez de la ejecución del software hecho con C++.

Para crear el encapsulado del mensaje CIP de acuerdo a lo que se describe en la sección 3.2.2, se utiliza una serie de estructuras definidas por el usuario que se encargan de recopilar los grupos de datos para finalmente formar el conjunto total de datos que formarán el mensaje CIP, estas estructuas se detallan a continuación:

1. CIPMsg: Esta estructura posee los dos campos principales del encapsulamento CIP, como son encapsulationHeader y commandSpecificData , el primero es una estrutura definida de usuario EncapHead, la segunda es un buffer de 256 bytes. Finalmente se tiene el campo de longitud que posee el tamaño en bytes de la estructura.

2. EncapHead: Contiene los campos de la cabeza del encapsulado, en este se encuentran los siete campos utilizados en esta sección del mensaje CIP.

3. CmdSpDataRegisterSession: Esta estructura es utilizada como el campo especifico de datos para el comando RegisterSession. Posee dos campos protocolVersion y optionFlags, que son igual a 1 y 0 respectivamente.

4. CmdSpDataSendRRdata: Esta estructura es utilizada como el campo especifico de datos para el comando SendRRData, contiene los cinco campos descritos para el comando, los campos typeId1 y typeId2 son estructuras especificas de usuario que contienen los datos especificos de este comando.

5. TypeId: Esta diseñado como un buffer de 9 bytes de longitud que contendrá el commando DF1 en el mensaje CIP, de acuerdo a lo explicado en parrafos anteriores.

6. TypeId1: Esta diseñado como un buffer de 12 bytes de longitud que contendrá la dirección IP en el mensaje CIP, de acuerdo a lo explicado en parrafos anteriores.

7. DF1Msg: Posee el commando DF1 a ser transmitido, se explicará en breve.

8. Tbuffer: Corresponde al buffer de transmisión que se utilizará para enviar el mensaje CIP a través de la conexión TCP, posee dos campos, un buffer de tipo char de tamaño 512 y el tamaño del buffer.

En la figura se muestra, la conformación del mensaje CIP de acuerdo a las estructuras descritas.

El comando DF1 enviado en el mensaje CIP se encarga de la lectura de datos de direcciones logicas dentro del PLC.

El espacio CMD tiene un valor de 0x0F y FNC 0xA2. STS es igual a 0x00 por defecto y TNS es un valor de continuidad que puede ser cualquier valor. ByteSize corresponde al tamaño en bytes de los datos que se quieren leer, comenzando en la posición descrita por los campos siguientes.

La memoria del PLC esa dividida en paginas las cuales son descritas por letras mayusculas en la parte inicial de la dirección (O,S,T,N,B). Cada una de estas páginas tiene un tamaño especifico y dentro de cada una pueden haber varios archivos los cuales están descritos por el dígito inmediatamente a la derecha del tipo de archivo a usar. Cada uno de los archivos está compuesto por 256 palabras de 16 bits cada una, el numero de palabra que se quiera usar esta dado por el digito a la derecha de los dos puntos (:). Finalmente es posible acceder a cualquiera de los 16 bits de la palabra por medio del ultimo digito a la derecha del backslash (/).

La posición de memoria en el comando DF1, están dados por el campo FileType que describe el tipo de pagina, asignando un número hexadecimal de acuerdo al tipo de página descrito en la dirección de memoria. FileNumber describe el numero de archivo y ElementNumber el numero de byte que se desea leer. La clase EthernetIp es capaz de manejar las direcciones de memoria de la misma forma en la que se acaba de describir, de modo que hace más fácil el direccionamiento de memoria.

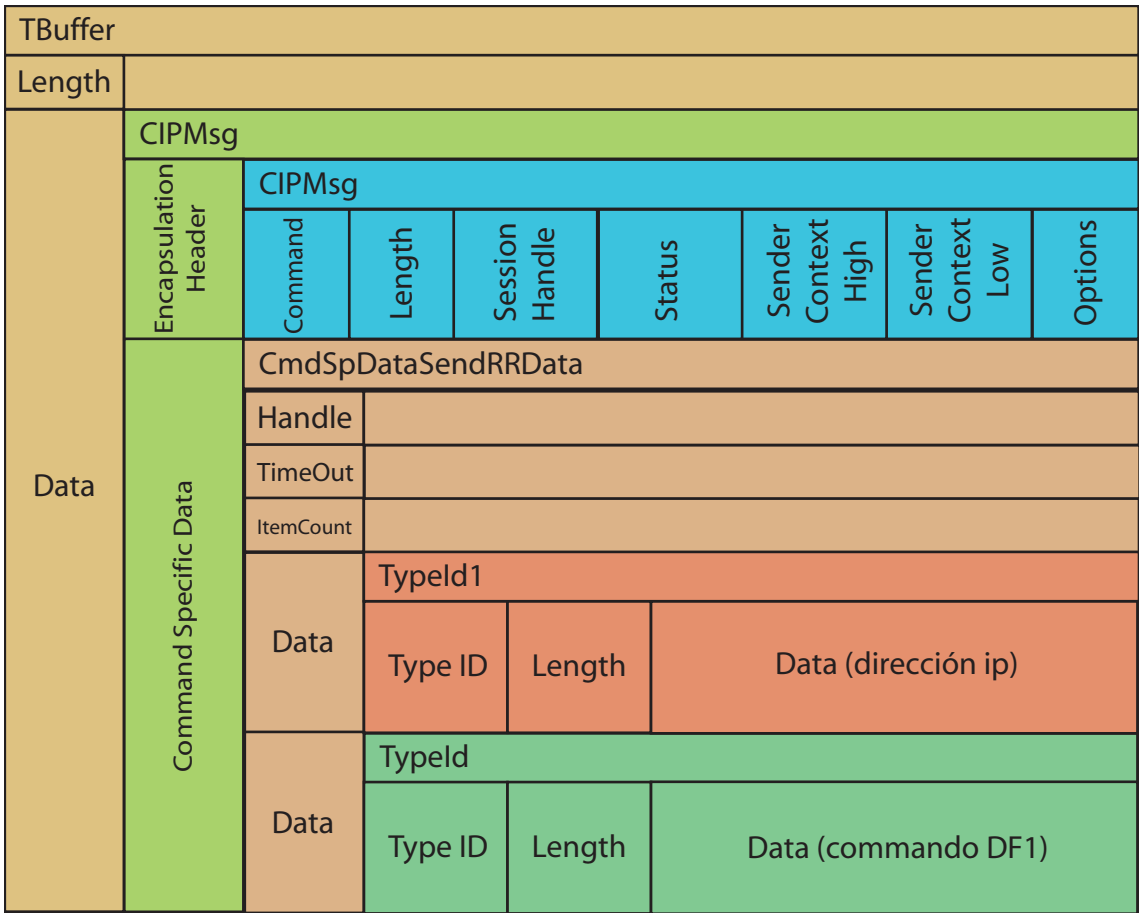


Figura 5.10: Encapsulamiento del mensajeCIP de acuerdo a las estructuras de usuario de la clase EthernetIP.

Para unir el código de C++ con el código en Java se hace uso de la herramienta SWIG. SWIG genera todo el código necesario para poder llamar las clases escritas en C++ desde Java y se encarga de toda la conversión de datos necesaria. El código generado por SWIG no se explicará aqui.

Para evitar la no compatibilidad y problemas de tipos que se puede presentar durante la conversión de tipos entre ambos lenguajes, se utilizó como salida para el comando readMemoryAddress, una cadena de caracteres (String), la cual esta conformada por una cadena de bits (unos y ceros) en ASCII, los cuales han sido formateados de acuerdo al resultado de la lectura de memoria en el PLC, cuando esta cadena es recibida por Java, es fácilmente manejable por medio de la libreria de manejo de caracteres de Java. El comando readMemoryAddress recibe la dirección IP del PLC donde se realizará la lectura, la posición de memoria a leer y la cantidad de bytes en la lectura a partir de la posición de memoria dada.

5.3.2 Herramienta de generación de reportes para LabMonitor

Como se menciono anteriormente la clase EthernetIP se creo como una librería para ser utilizada desde una aplicación en Java para poder realizar lecturas de memoria en el PLC y posteriormente generar los reportes de fallas que ya se han mencionado antes.

Para realizar esta tarea de forma ordenada se implementó una base de datos MySQL en la cual se tienen almacenados todos los datos necesarios, como son direcciones IP, posiciones de memoria, mensajes de texto y otros adicionales. En la figura 5.11 se muestra la estructura de la base de datos "Ransco" creada con MySQL.

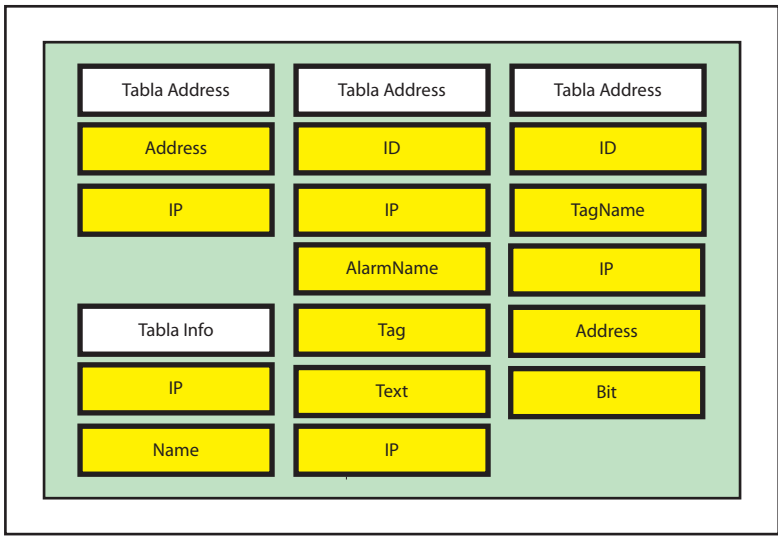


Figura 5.11: Estructura de la base de datos para la configuración de la herramienta de reportes de LabMonitor.

Esta base de datos se encuentra dividida en 4 tablas principales. La primera tabla "info" posee los datos del nombre y la dirección IP definida para cada una de las interfaces 1761 NET-ENI. Los nombres dados a cada cámara deben coincidir con los nombres dados a las cámaras en la base de datos de LabMonitor. El formato de la dirección IP es el formato estándar de cuatro números de 0 a 255 separados por puntos. La tabla "address" contiene las direcciones de memoria que son de importancia en cada una de las cámaras, donde cada cámara esta indentificada por su dirección IP. La tabla "tags" es similar a la estructura que utiliza el programa de aplicación del panel táctil para agrupar las posiciones de memoria. Cada uno de los tags tiene asociado una posición de memoria y además un número de bit, de forma que cada uno brinda toda la información necesaria para poder direccionar la memoria y extraer la información que se requiere de los resultados de la lectura. Finalmente la tabla "alarms" posee la información de los mensaje de texto de las distintas alarmas. A cada una de los mensajes se le asigna uno de los tags de forma que

se tenga la información de cual es el aquel dato que al revisarlo provoque que se dispare la alarma.

5.3.3 Software para el hardware de adquisición de datos

Como se mencionó anteriormente el módulo ez80f91 sirve como servidor para la adquisición y envío de muestras de la señal de vibración. El cliente remoto solicita paquetes de muestras de acuerdo a un protocolo establecido. La cantidad de muestras que conforman un paquete pueden ser configurados en el módulo de acuerdo a las necesidades de la aplicación (esta funcionalidad aun no ha sido implementada en el prototipo final, se espera que la configuración se haga por medio de una página web aprovechando la funcionalidad de servidor HTTP que posee el módulo ez8f91).

El software del módulo contiene básicamente cinco partes:

1. La configuración de red para el módulo en donde se configuran los parámetros de red del módulo (dirección IP, máscara de red, puerta de salida), así como los protocolos de red que serán utilizadosñ para esto la programación del módulo permite agregar los protocolos de red que se necesiten según la aplicación, de forma que se pueden agregar protocolos como ARP, HTTP, DNS, TELNET, TCP, FTP, STMP y otros, claro está que es necesario agregar los protocolos que son necesarios para poder brindar el mínimo de funciones de red posibles, además otros protocolos necesitan de otros protocolos para poder funcionar. La configuración de red no se explicará en acá, si se requiere alguna referencia al respecto puede remitirse a [10].

2. El servidor TCP, el cual espera por la conexión de un cliente a través del puerto 5000.

3. La configuración del hardware de adquisición de datos, en esta parte se detalla cuales son los parámetros necesarios para poder realizar la recolección y transmisión de los paquetes de muestras. Además es acá donde se configura la comunicación SPI en el módulo ez80f91.

4. El protocolo de adquisición de datos en donde se reciben las solicitudes, se recolectan las muestras y finalmente se envían los paquetes de muestras.

5. La herramienta de muestreo que se ejecuta la tarea de solicitar las muestras del ADC por medio del envío de comandos a través del puerto SPI.

En la tabla 5.1 se muestran las principales funciones del software del módulo de adquisición de datos.

Tabla 5.1: Funciones del software del hardware adquisición de datos

Función	Descripción
void ez80_server(void)	Inicia el servidor TCP y espera por una conexión
void process_connection(DID TCPConnDevID)	Procesa la conexión de un cliente, contiene el protocolo de adquisición de datos
unsigned char adc_readchar (char channel)*	Obtiene una muestra del ADC por medio de la comunicación SPI
void spi_init(void)	Inicializa el puerto SPI en el módulo ez80f91
void hardware_test(DID TCPConnDevID)	Ejecuta una prueba del hardware de adquisición de datos y envía los resultados al cliente
void load_hardware_configuration(void)	Lee el archivo de configuración del módulo

Luego de inicializar los parámetros de red del módulo el servidor TCP se incializa y entra en modo de espera por la conexión de un cliente. Cuando esto finalmente se da, se inicia el procesamiento de la conexión por medio de la función process_connection, en la cual se desarrolla el protocolo de adquisición de datos.

Al iniciar la conexión se realiza la inicialización del canal de comunicación SPI por medio de la función spi_init, esta se encarga de configurar el puerto y los registros SPI del módulo ez80f91 para que este funcione como maestro durante las secciones de tranmisión (para una mayor referencia de esta configuración [10]), luego se ejecuta la lectura del archivo de configuración del hardware de adquisición de datos, esto inicializa los valores correspondientes a la cantidad de muestras que se deben de recolectar para cada uno de los canales del ADC, así como cuales canales se encuentran habilitados (esta configuración se realizará por medio de una página web que se implementará en el futuro), para luego ejecutar una prueba sobre el muestreo del ADC según los canales que se encuentren habilitados. Para esto se realiza un pequeño muestreo sobre las señales disponibles y se hace una revisión de los valores obtenidos , al final de la muestra los valores son recopilados y enviados como una cadena de caracteres donde se indica el estado de disponibilidad y funcionamiento de cada canal al cliente, de forma que esta información sea tomada en cuenta en el caso de fallas y mantenimiento.

Una vez realizados los procedimientos de inicialización, el procesamiento de la conexión continua con la espera de comandos para la solicitud de muestras. Para esto el cliente envía un paquete donde se indica la cantidad de muestra y el canal del cual se deben tomar las muestras, el paquete además posee un campo correspondiente al número de solicitud el cual debe ser devuelto al cliente como indicador de respuesta para cada solicitud de paquetes. La respuesta tendrá el mismo encabezado que la solicitud y al final tendrá

los paquetes de muestras solicitadas. Debido a la implementación de la pila TCP de Zilog, solamente cadenas de caracteres de tipo byte pueden ser enviados a través de la conexión TCP del módulo, dado que las muestras obtenidas del ADC son valores de 12 bits, es necesario dividir cada una de las muestras en parte alta primero y parte baja después, por lo que si se envían 256 caracteres en el campo de datos de la respuesta, esto corresponderá a 128 muestras de 12 bits, las cuales tendrán que ser reensambladas por el cliente.

En el siguiente código se muestra el procedimiento de adquisición de una muestra al ADC

```
1    PB_DR &= 0xEF;
2    for(i = 0; i ¡ 6; i++);
3    aux = channel ≫ 2;
4    aux =0x06;
5    SPI_TSR = aux;
5    tmp = SPI_SR;
6    tmp &= 0x80; 7      while(tmp == 0){
8            tmp = SPI_SR;
9            tmp &= 0x80;
10   }
11   aux = channel ≪ 6;
12   SPI_TSR= aux;
13   tmp = SPI_SR;
14   tmp &= 0x80;
15   while(tmp == 0){
16           tmp = SPI_SR;
17           tmp &= 0x80;
18   }
19   data_readH = SPI_RBR;
20   SPI_TSR = 0x00;
21   tmp = SPI_SR;
22   tmp &= 0x80;
23   while(tmp == 0){
24           tmp = SPI_SR;
25           tmp &= 0x80;
26   }
27   data_readL = SPI_RBR;
28   PB_DR —= 0x10;
29   data_readH &= 0x0F;
30   output[1] = data_readH;
31   output[0] = data_readL;
```

El proceso inicia poniendo la señal chip_select en bajo la cual habilita el ADC está señal está controlada por medio del bit 5 del puerto PB (línea 1). Para realizar la comunicación con el ADC, esto se realiza con el envío de la configuración del ADC primeramente, para esto se envía un primer byte en donde los 3 bits menos significativos corresponden a un bit de inicio, un bit de configuración del canal donde se indica si este muestrea la señal como un canal diferencial (con respecto a la resta de dos de los canales del ADC) o bien un canal simple (con respecto al canal y a tierra en el ADC) y luego se envía el tercero

(D2) de tres bits que seleccionan uno de los 8 canales del ADC, el resto de los bits de este primer byte se ponen como cero, en el caso de la aplicación de la solución, se utiliza un canal de muestreo simple y el canal se escoge de acuerdo al valor enviado en la solicitud del cliente, lo que requiere que se de formato a los bits del primer byte enviado al ADC para extraer el primer bit del valor del canal indicado (líneas 3-4). El segundo byte contiene en sus dos bits más significativos los otros bits que indican el canal seleccionado (D1, D0), los otros bits del byte no son importantes, para esto de nuevo es necesario dar formato al byte antes de enviarlo (línea 13).

Como respuesta a estos comandos se el ADC envía dos bytes de respuesta con el valor de la muestra obtenida, estos dos bytes son obtenidos del registro de datos SPI del módulo ez80f91, para esto es necesario realizar un periodo de espera para la carga del registro (líneas 9-11, 17-19) para luego realizar la lectura de los datos (líneas 21, 29). Finalmente los datos son retornados por la función en un arreglo de dos bytes (líneas 32-33) para poder ser separados y enviados como parte alta y baja del valor muestreado.

El proceso termina cuando el cliente envía la solicitud de cierre de conexión, lo que pondrá al servidor a la espera de un nuevo cliente.

5.3.4 Herramienta de reconocimiento de condición

La herramienta de reconocimiento de condición tiene la función de ejecutar el proceso de discretización y reconocimiento de una señal continua de vibración (continua se toma acá dentro de los parámetros discretos que tiene una señal muestreada), el propósito es la extracción de características de una señal de evaluación que permitan reconocer la fuente a la cual esta pertenece, estas fuentes son estados conocidos del equipo en el cual se esta aplicando la herramienta. Es sabido que conforme cambia el estado de un equipo en particular se presentan nuevos rasgos en la señal de vibración que este produce (o bien en algún otro tipo de señal que refleje estás características), de esta forma, un mismo equipo se puede analizar como distintas fuentes que producen una señal con características distintas en función del estado que presenten. De esta forma recolectando las características y luego aplicando algoritmos de reconocimiento, como es el caso de los Modelos Ocultos de Markov, es posible clasificar una la secuencia que se está evaluando.

Las caraterísticas principales de la herramienta de reconocimiento de condición son:

1. La herramienta está desarrollada enteramente en Python, el cual es un lenguaje interpretado de amplio uso a nivel comercial y que, como todo lenguaje de este tipo, permite el desarrollo de aplicaciones de forma muy agil, de forma que se deja el trabajo pesado a otros lenguajes como C ó C++. El uso de Python se ha difundido

mucho a nivel de cálculo computacional, especialmente en librerías científicas, debido a que Python permite un manejo muy fácil de los datos como variables y arreglos.

2. La herramienta está conformada por cinco clases las cuales desarrollan una labor específica dentro del proceso de clasificación. El uso de clases que permite Python, posibilita un manejo más fácil de los procedimientos y el manejo de variables en la aplicación final, sin dejar de lado la reutilización de clases.

3. La herramienta hace uso de un grupo de librerías de código abierto que brindan el soporte de cálculo principal de la aplicación. Las principales librerías son escritas en C y poseen una interfaz (wrapper) para Python que permite su utilización en este lenguaje.

4. La herramienta esta basada en el procedimiento descrito en el artículo "Hidden Markov Model-based Tool Wear Monitoring In Turning" a cargo de Litao Wang, Mostafa G. Mehradi y Elijah Kannatey-Asibu, Jr.

Directorios y archivos

Primeramente se dará un vistazo a la estructura de directorios y archivos que utiliza la herramienta de clasificación.

De acuerdo a lo explicado en la sección 5.1.2 y utilizando como referencia la figura , la aplicación utiliza varios tipos de datos: las muestras de la señal de vibración, los coeficientes wavelets, los vectores de características, los archivos de secuencias, los codebooks, los HMMs y finalmente los valores de probabilidad obtenidos al final del proceso.

El directorio "data" recopila los archivos de muestras que se obtuvieron durante el proceso de muestreo del hardware de adquisición de datos. Los archivos contenidos en este directorio son de tipo ".dat", los datos están distribuidos seguidos uno del otro separados por un retorno de línea. El nombre de cada archivo esta formado por la dirección IP del servidor del cual se obtuvieron las muestras, el número de canal y un número de identificación.

En el directorio "vector_features" se depositan los vectores de características que se obtuvieron luego de obtener lo wavelets y las energías de las muestras de la señal. Dentro de este se encuentran los directorios "training" y "evaluation" , el primero almacena los vectores de características que se utilizan para el entrenamiento de los codebooks y de los HMMs, acá se dividen los datos de acuerdo al servidor y canal de donde se obtuvieron las muestras, el directorio "evaluation" posee los vectores que serán evaluados por la herramienta. Cabe indicar que en el directorio "training" son acumulados todos los datos que han sido utilizados para el entrenamiento, esto para poder realizar el re-entrenamiento

de los codebooks y de los HMMs. Los archivos contenidos en "vector_features" utilizan el formato "CSV" (Comma Separated Values), este tipo de archivos son utilizados ampliamente para aplicaciones que usan grandes cantidades de datos en columnas o vectores como es el caso de Microsoft Excel, en estos archivos los componentes de los vectores están separados por comas (",") y los distintos vectores a su vez están separados por retornos de línea. El uso de estos archivo, como se verá más adelante, permitirá que los vectores sean visualizados por medio de la herramienta HiSee. Cada archivo tiene un nombre característico dado por el usuario para identificar la fuente de los datos.

En el directorio "vector_quantization" se almacenan los codebooks y los archivos de codewords de las secuencias evaluadas con los distintos codebooks. Los codebooks son archivos XML, los cuales son muy usados en la herramienta. Los archivos XML son utilizados en una gran cantidad de aplicaciones y en distintas aplicaciones para almacenamiento de datos, esto debido a que son muy fáciles de leer, son independientes de la aplicación, son extremadamente amplios y son fácilmente interpretables por el usuario. Existe una gran cantidad de información sobre el formato XML en muchas fuentes distintas. En esta la raíz de los datos la indica la etiqueta "codebook", cada uno de los codewords miembros del codebook son almacenados dentro de una etiqueta "codeword" y dentro de esta se almacena cada uno de los componentes de los vectores en una etiqueta "component", la cantidad de miembros en cada codebook dependerá de la dimensión de los vectores de características con los que se este trabajando. Cada codebook está identificado por la dirección IP y el número de canal correspondiente a los datos. En el directorio "codewords" están los resultados de la evaluación de las secuencias contra el codebook correspondiente. Estos archivos son scripts de Python con el formato necesario para que sean cargados por la librería GHMM para el entrenamiento o evaluación de los HMMs.

El directorio "hmm" posee los HMMs entrenados así como los archivos de probabilidades que son resultado de la evaluación de secuencias con los HMMs. En ambos casos se utilizan archivos XML. Los HMMs son almacenados en archivos nombrados de forma característica con un nombre dado por el usuario para identificar la fuente de los datos, esto ayudará a identificar la procedencia de las probabilidades calculadas. Cada archivo tiene como raíz la etiqueta "hmm", dentro de esta se encuentran separados cada estado con una etiqueta "state", a su vez dentro de esta se encuentran: los valores de probabilidad inicial del estado dado por la etiqueta "initial"(π), los valores de probabilidad de observación, dados por la etiqueta "output" (B) para ese estado y finalmente los valores de probabilidad de transición a los otros estados con la etiqueta "transition" (A). Todos estos dados agrupados forman entonces las matrices A, B y π para cada HMM entrenada. Los archivos de probabilidades almacenan los valores de las probabilidades que son resultado de aplicar las secuencias de evaluación contra los HMM.

En el directorio "wavelets_coefficients" se almacenan los coeficientes wavelets que han sido

Tabla 5.2: Procedimientos de la clase FileHandler

Procedimiento	Descripción
def __init__(self, filename, op)	Constructor de la clase recibe el nombre de archivo y la operación a realizar sobre el archivo escritura ("w") o bien lectura ("r"), abre el archivo especificado
def write(self, data)	Escribe el vector data en el archivo especificado
def read(self)	Retorna un vector con el contenido del archivo especificado
def write_csv(self, data)	Escribe el vector data en el archivo CSV especificado
def read_csv(self)	Retorna un vector con el contenido del archivo CSV especificado
def close(self)	Cierra el archivo

calculados en el proceso de discretización, esto en caso de que se requiere alguna revisión por parte del usuario. Los archivos están identificados con la dirección IP del servidor de datos, el canal donde se tomaron las muestras y un valor de identificación, la extensión de los archivos es ".wcd".

Clase FileHandler

Esta clase se encarga de la escritura y lectura de datos en los archivos utilizados por la herramienta, principalmente los archivos ".dat", ".csv" y ".wcd".

En la tabla 5.2 se muestran los procedimientos que forman parte de esta clase

Clase SocketHandler

En esta clase se realiza la comunicación con el servidor del hardware de adquisición de datos, esto por medio de un socket TCP en la dirección IP y número de puerto especificados.

En la tabla 5.3 se muestran los procedimientos de la clase SocketHandler

Clase DSPHandler

En esta clase se obtienen los vectores de características a partir de las muestras de la señal de vibración, esto obteniendo la DWT de las muestras y luego calculando la energía normalizada de cada banda de frecuencia, lo que producirá una secuencia vectorial representativa de la señal, como se detalla en la figura 5.1.2 de la sección 5.3.4.

Tabla 5.3: Procedimientos de la clase SocketHandler

Procedimiento	Descripción
def __init__(self, host, port, channel)	Constructor de la clase recibe los parámetros de red, dirección IP y número de puerto, así como el canal en donde se realizará el muestreo de datos
def connect(self)	Abre un socket TCP en la dirección IP y número de puertos especificados
def get_hardware_test(self)	Recibe y procesa la prueba de hardware que recibe del servidor del hardware de adquisición de datos, es el primer paso en el procedimiento de conexión luego de abierto el socket, en caso de no recibirlo se eleva una excepción
def get_data_packets(self, packets, id)	Este procedimiento se encarga de enviar una solicitud de muestras al servidor del hardware de adquisición de datos con el número de paquetes especificados, al recibirlos formatea los bytes de partes alta y baja para formar un único valor. Las muestras obtenidas son almacenadas en un archivo ".dat" con el identificador provisto por medio de una instancia de la clase FileHandler
def close(self)	Cierra el socket

Para el cálculo de la DWT, la clase DSPHandler hace uso de la Librería científica de GNU (GSL -GNU Scientific Library) [12]. Esta es una librería numérica para programadores de C, C++ y Python de software libre bajo la licencia pública general GNU. La GSL provee una gran cantidad de rutinas matemáticas como generadores de números aleatorios, integrales, transformadas, métodos númericos y muchas otras. Existe alrededor de 1000 funciones implementadas en la librería.

Junto con esta librería es necesario el uso de una interfaz que permita utilizar la librería desde Python, para esto el proyecto cuenta la librería PyGSL (ver [26]), la cual es una colección de implementaciones de la librería GSL en Python. Posee gran cantidad de las funciones que ofrece GSL y se encuentra en desarrollo de nuevas funciones. Para el cálculo de la DWT, PyGSL ofrece una implementación identica a la ofrecida por GSL.

Para el cálculo de la DWT se utilizo como madre wavelet el wavelet de Daubechies de orden 4, esto debido a que esta familia de wavelet se utiliza comúnmente para el reconocimiento de patrones y procesamiento de señales por la capacidad que está posee en obtener grandes niveles de detalle, junto con esto se suelen utilizar ordenes de detalle bajos en este tipo de aplicaciones. En el siguiente código de Python muestra el cálculo de los coeficientes wavelet para un grupo de muestras de la señal de vibración

```
1   energies = []
2   for packet in range(self.packets):
3           reads signal samples from file
4           filename = "./data/" + self.host + "/" + self.channel + "/" + str(packet) + ".dat"
5           fileReader = FileHandler(filename, "r")
6           data = fileReader.read()
7           fileReader.close()
8          computes wavelet coefficients
9          l = len(data)
10          w = wavelet.daubechies(4)
11          ws = wavelet.workspace(l)
12          wc = w.transform_forward(data,ws)
13          store wavelet coefficients
14          filename = "./wavelet_coefficients/" + self.host + "/" + self.channel + "/" + str(packet) + ".wcd"
15          fileWriter = FileHandler(filename, "w")
16          fileWriter.write(wc)
17          fileWriter.close()
18          computes normalized energies of frequency bands
19          energies.append(self.normalize_energies (self.get_energy_bands(wc)))
20          creates dir for vectorial sequences
21          path = "./vector_features/" + nature + "/" + self.host + "/" + self.channel + "/"
22          if not access(path, F_OK):
23                  mkdir(path)
24          stores vectorial sequence
25          filename = name + ".csv"
26          fileWriter = FileHandler(path + filename,"w")
27          fileWriter.write_csv(energies)
28          fileWriter.close()
```

El proceso consiste en cargar cada uno de los grupos de las muestras que se encuentran en el directorio "data" (líneas 4-7), para esto se crea una instancia de la clase FileHandler para realizar la lectura de cada archivo, de esta forma la longitud de la secuencia del vector de características final, dependerá de la cantidad de grupos de datos obtenidos del directorio "data".

En la líneas 9-12 se muestra el cálculo de los coeficientes wavelet, para esto es necesario crear una instancia del tipo de wavelet que se va a crear, como en el caso de la herramienta se utiliza Daubechies de orden 4 (línea 10), luego es necesario crear un espacio de memoria en donde se almacenarán temporalmente los coeficientes y a partir del cual se realizarán

los cálculos. Finalmente se aplica la DWT a las muestras (línea 11) y se obtiene un vector con los coeficientes wavelet. En este vector se encuentran todos los niveles de coeficientes calculados para la muestra de datos. Para esto dado el número de muestras n, el cual es una potencia de dos, se tiene entonces $J = log\_2(n)$ niveles de coeficientes, de esta forma los niveles están dador por $j = 0 \cdots J - 1$, dentro de cada nivel los coeficientes están indexados como $k = 0 \cdots 2^j - 1$, así cada nivel tiene al menos S^j muestras. Con esto en mente la conformación del archivo de salida de la DWT esta dado de la siguiente forma

$$(s_{-10}, d_{00}, d_{10}, d_{11}, d_{20}, d_{21}, d_{22}, d_{23}, \cdots, d_{jk}) \tag{5.2}$$

Esta configuración debe ser tomada en cuenta para separar los coeficientes wavelet y realizar el cálculo de la energía, en el siguiente código se muestra el procedimiento para el cálculo de la energía tomando en cuenta la configuración anterior

```
1   energies = []
2   count = 1
3   for j in range(self.scales):
4       energy = 0
5       for k in range(pow(2, j)):
6           energy = energy + pow(array[count], 2)
7           count += 1
8           energy = energy / pow(2, j)
9           energies.append(energy)
10  return energies
```

Finalmente se calculan los valores de energía normalizados (línea 19) y se almacenan los vectores resultantes en un archivo CSV (líneas 25-27)

En la tabla 5.4 se muestran los procedimientos de la clase DSPHandler.

Clase CodebookHandler

La clase CodebookHandler se encarga del entrenamiento de los codebooks de la herramienta, así de evaluar una secuencia de entrada contra alguno de los codebooks almacenados para pasar de una secuencia de tipo vectorial a una secuencia discreta que sea aplicable a los HMMs.

Tabla 5.4: Procedimientos de la clase DSPHandler

Procedimiento	Descripción
def __init__(self, host, channel, packets, scales)	Constructor de la clase recibe la dirección IP y el canal en donde se realizó el muestreo de datos, recibe además el número de paquetes depositados en el directorio "data" y el número de scalas a usar en el cálculo de las energías normalizadas
def get_wavelet_coef(self, name, nature)	Realiza el cálculo de los coeficientes wavelet por medio de la DWT de Daubechies de orden 4 y genera el archivo de características obtenido a partir de todos los paquetes de datos contenidos en el directorio "data", recibe el nombre característico para el archivo CSV y una indicación del directorio de "vector_features" donde se almacenará el archivo("training" o "evaluation")
def get_energy_bands(self, array)	Calcula la energía de las bandas de energía del vector de entrada Array
def normalize_energies(self, energies)	Normaliza el vector de energías

La clase tiene una implementación del algoritmo LBG (el cual se detalla en la sección 3.2.6) realizada por el estudiante, la cual almacena el codebook resultante en un archivo XML.

La evaluación de las secuencias contra el codebook se realiza por medio del cálculo de la distancia euclideana entre cada miembro de la secuencia y cada miembro del codebook respectivo, aquel miembro del codebook que tenga la menor distancia al miembro de la secuencia será aquel cuyo índice será utilizado para representar al miembro de la secuencia en la secuencia discreta final.

En la tabla 5.5 se muestran los procedimientos de la clase CodebookHandler.

Clase GhmmHandler

En esta clase se realiza el entrenamiento de los HMMs y la evaluación de las secuencias discretas contra estos.

En la tabla 5.6 se muestran los procedimientos de la clase GhmmHandler.

La clase hace uso de la librería GHMM (ver GHMM), la cual es una librería de código abierto dedicada a la implementación de los HMMs, contiene todos los procedimientos necesarios para la evaluación, entrenamiento y estudio de los HMMs, junto con esto el proyecto posee una interfaz para Python que permite su fácil uso desde este lenguaje.

En el siguiente código se muestra el procedimiento de entrenamiento que se utiliza para

Tabla 5.5: Procedimientos de la clase CodebookHandler

Procedimiento	Descripción
def __init__(self, host, channel, dimension)	Constructor de la clase recibe la dirección IP y el canal en donde se realizó el muestreo de datos, recibe además la dimensión de los vectores de la secuencia
def lbg_algorithm(self, codebooksize, error)	Realiza el entrenamiento del codebook, recibe el tamaño del codebook el cual debe ser un número par y no debe ser igual al número de miembros de la secuencia, además recibe el error tomado en los cálculos
def get_codeword_set(self, data)	Obtiene la secuencia discreta para una secuencia vectorial de evaluación a partir del cálculo de la distancia euclideana entre los miembros de la secuencia de evaluación y los miembros del codebook
def write_xml(self, codewords, host, channel)	Crea un archivo XML donde se deposita los codewords del codebook calculado
def read_xml(self, host, channel)	Lee un archivo codebook XML
def create_data_script(self, nature, data)	Crea un script con formato de Python en donde se almacenan las secuencias de evaluación o entrenamiento según lo indica nature, el archivo es importado en la clase GhmmHandler

Tabla 5.6: Procedimientos de la clase GhmmHandler

Procedimiento	Descripción
def __init__(self, N, M, host, channel)	Constructor de la clase recibe la dirección IP y el canal en donde se realizó el muestreo de datos, recibe además la longitud del alfabeto y la cantidad de estados del modelo
def view_hmm(self)	imprime el modelo actual en pantalla
def train_hmm(self, name, A, B, pi)	Entrena un HMM en base a una secuencia de entrenamiento y condiciones iniciales
def evaluate_sequence(self)	Evalua una secuencia con respecto a los modelos existentes de un servidor en particular y obtiene las probabilidades resultantes
def get_hmm(self)	Obtiene los parámetros A, B y pi para el HMM actual
def write_xml(self, filename)	Almacena un HMM en un archivo XML
def read_xml(self, filename)	Lee un HMM de un archivo XML

crear nuevos HMMs a partir de datos conocidos

```
1    initialize model
2    self.A = A
3    self.B = B
4    self.pi = pi
5    self.m = HMMFromMatrices(self.sigma, DiscreteDistribution(self.sigma), self.A, self.B,self.pi)
6    loads trainning sequence
7    module = __import__('vector_quantization. codewords.' + self.host + '_' + self.channel, globals(), locals(),["])
8    train hmm usgin Baulm-Welch algorithm
9    self.m.baumWelch(module.train_seq)
10    stores new hmm
11    self.write_xml(name)
```

El procedimiento requiere que se ingrese condiciones iniciales (líneas 2-4), dado que para el entrenamiento se utiliza el algoritmo de Baum-Welch. Una vez obtenidas las condiciones iniciales se crea un HMM (línea 5) el cual será reestimado por el entrenamiento para el reconocimiento de las secuencias de entrenamiento, las cuales son importadas a la aplicación (línea 7) por medio de la secuencia __import__, que es un procedimiento de Python que permite importar scripts de Python durante ejecución, lo que resulta muy útil para gran variedad de aplicaciones. Finalmente se hace uso de la librería GHMM para reestimar el modelo con la secuencia de entrenamiento (línea 9), y una vez que se obtiene el nuevo modelo este es almacenado en un archivo XML (línea 11).

Para la evaluación de una secuencia se tiene el siguiente código de Python

```
2    path = "./hmm/hmms/" + self.host + "_" + self.channel + "/"
3    hmms = listdir(path)
4    probs = []
5    for hmm in hmms:
7            self.read_xml(hmm)
9            filename = "vector_quantization.codebooks ." + self.host + "_" + self.channel
10             from filename import test_seq
12             v = self.m.viterbi(test_seq)
13             probs.append(v[1])
15   probHandler = ProbHandler(self.host, self.channel)
16   result = probHandler.add_probability_ data(probs)
```

Primero se obtienen los nombres de todos los modelos que se encuentran disponibles para el servidor de interés (líneas 2-3) cada uno de los HMMs que se encuentran en el directorio es leído (línea 7), luego se carga la secuencia de evaluación (línea 9) y finalmente se calcula la probabilidad de la secuencia dado cada modelo en particular (líneas 12-13), los resultados son acumulados para ser procesados en una archivo de probabilidades XML (líneas 15-16).

Capítulo 6

Análisis de resultados

6.1 Resultados experimentales

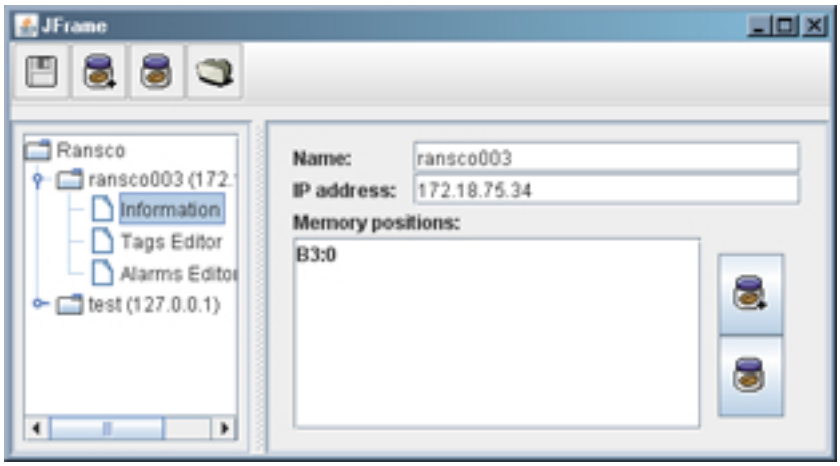


Figura 6.1: Interfaz de configuración de red para la herramienta de reportes de fallas de Lab-Monitor.

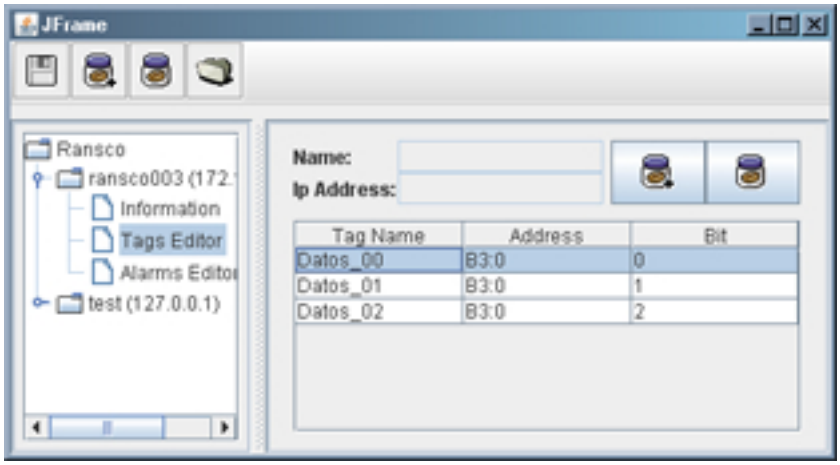


Figura 6.2: Interfaz de configuración de las posiciones de memoria para la herramienta de reportes de fallas de LabMonitor.

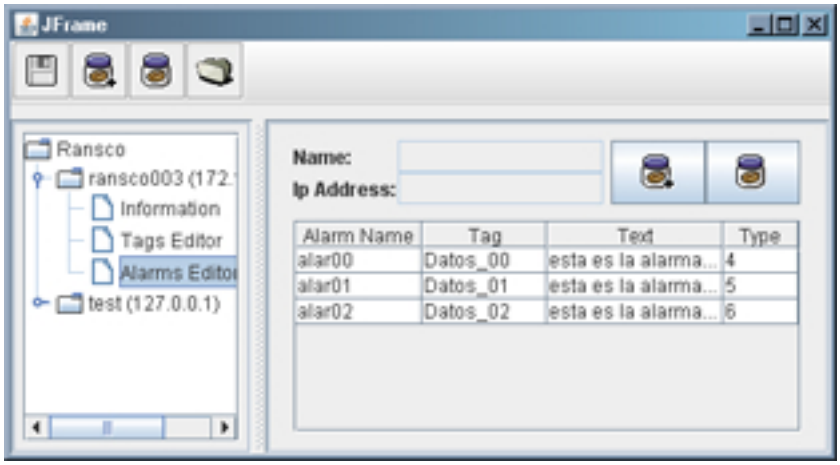
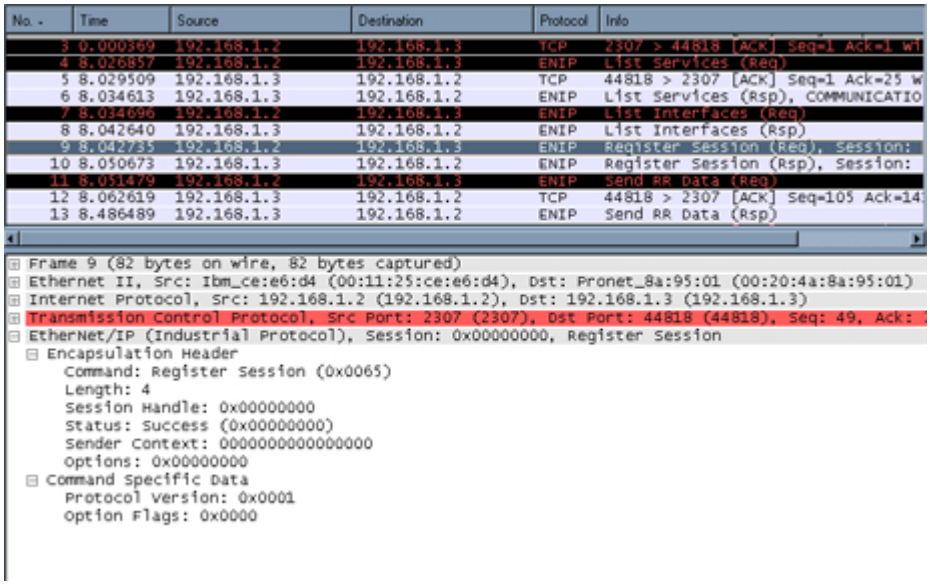
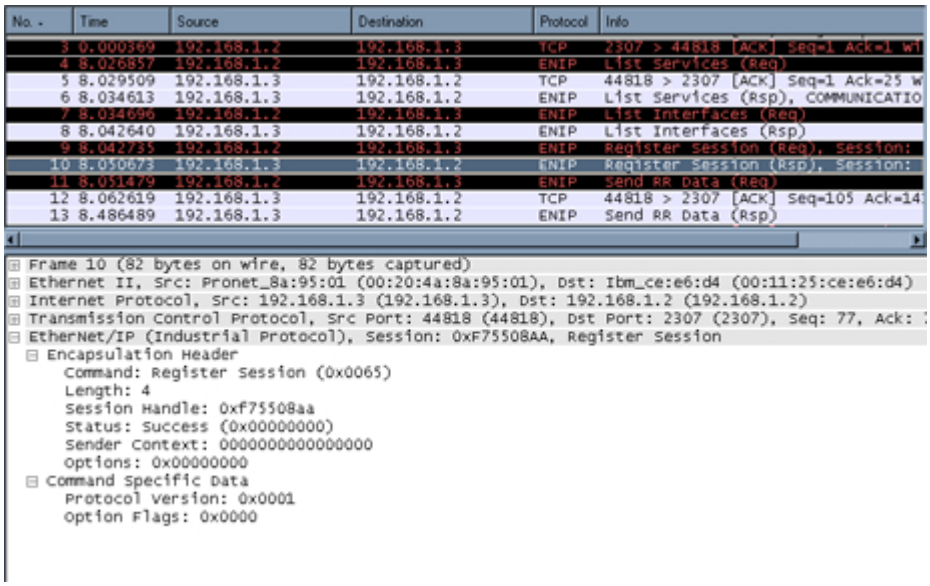


Figura 6.3: Interfaz de configuración de los mensajes para la herramienta de reportes de fallas de LabMonitor.

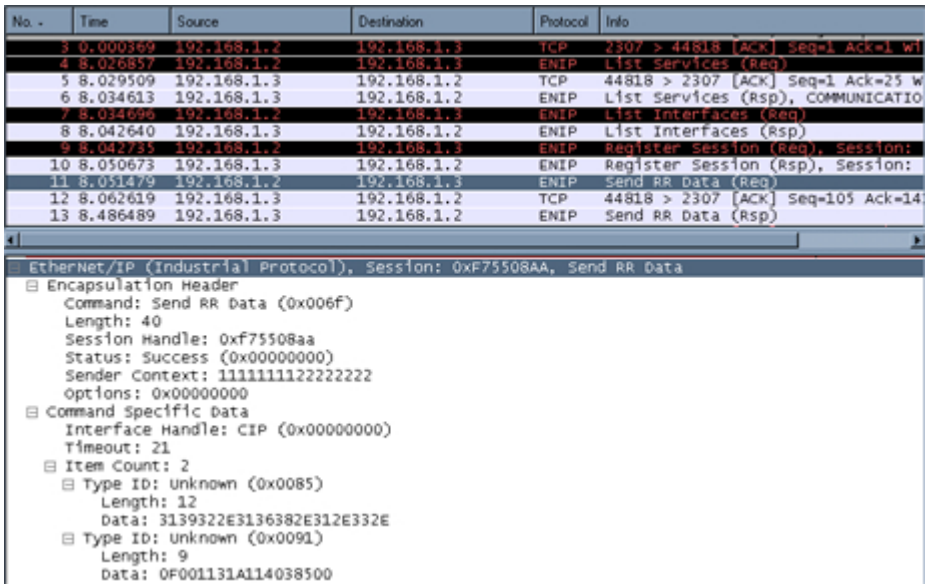


(a)

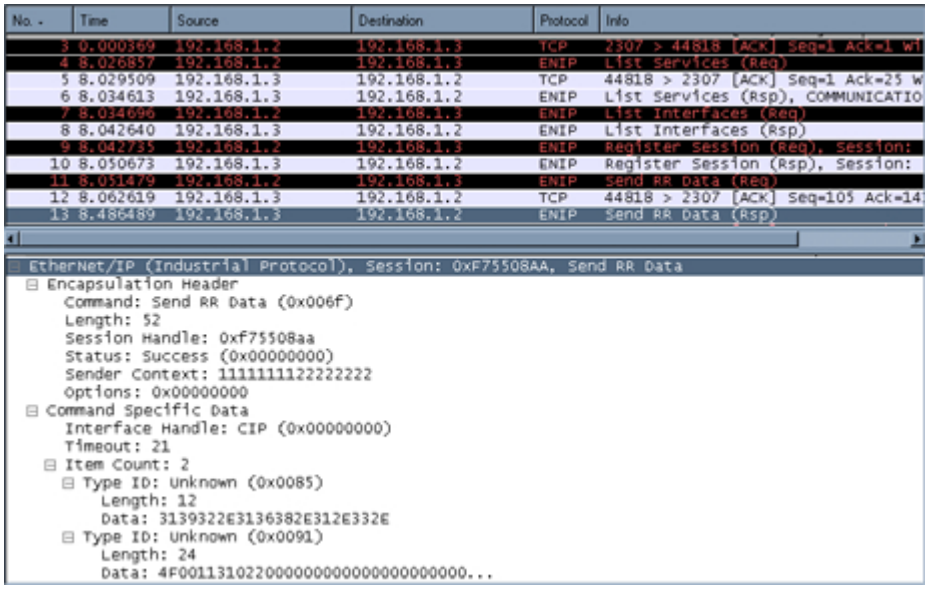


(b)

Figura 6.4: Captura de paquetes de red (a) para la solicitud de registro de sesión y (b) la respuesta para esta solicitud.

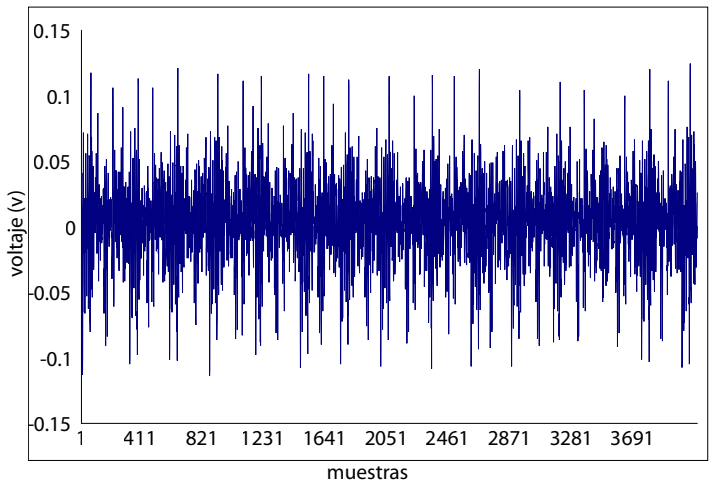


(a)

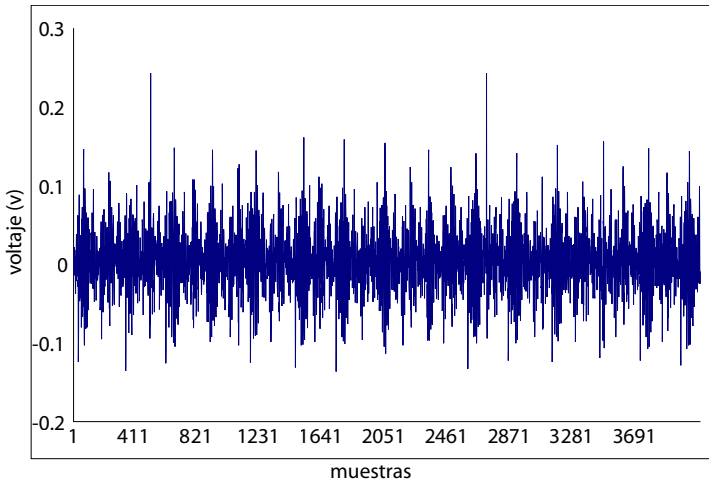


(b)

Figura 6.5: Captura de paquetes de red (a) para la solicitud para la lectura de memoria y (b) la respuesta para esta solicitud.

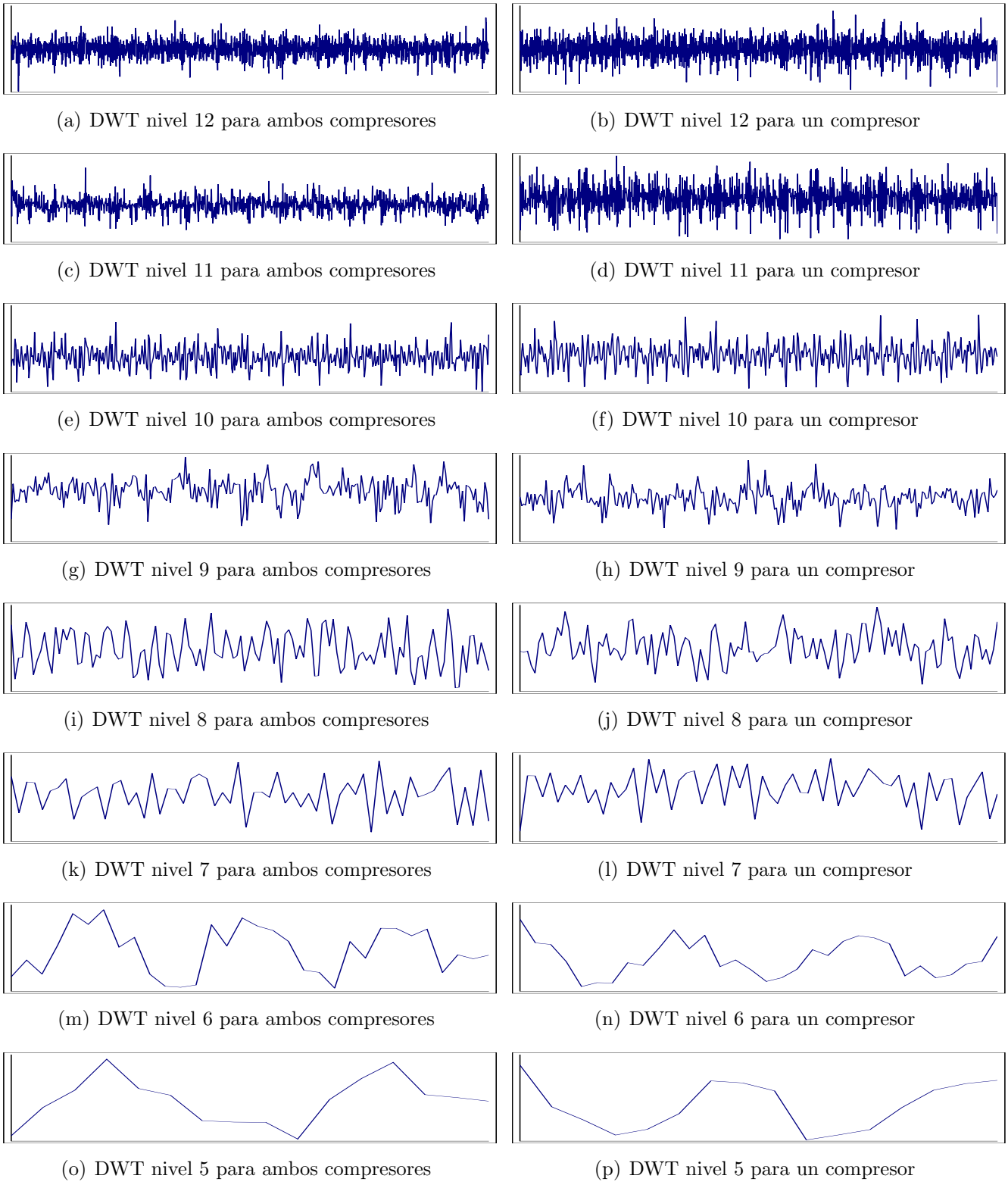


(a)



(b)

Figura 6.6: Señales de vibración experimentales (a) para ambos compresores en funcionamiento y (b) para un solo compresor en funcionamiento.



(a) DWT nivel 12 para ambos compresores

(b) DWT nivel 12 para un compresor

(c) DWT nivel 11 para ambos compresores

(d) DWT nivel 11 para un compresor

(e) DWT nivel 10 para ambos compresores

(f) DWT nivel 10 para un compresor

(g) DWT nivel 9 para ambos compresores

(h) DWT nivel 9 para un compresor

(i) DWT nivel 8 para ambos compresores

(j) DWT nivel 8 para un compresor

(k) DWT nivel 7 para ambos compresores

(l) DWT nivel 7 para un compresor

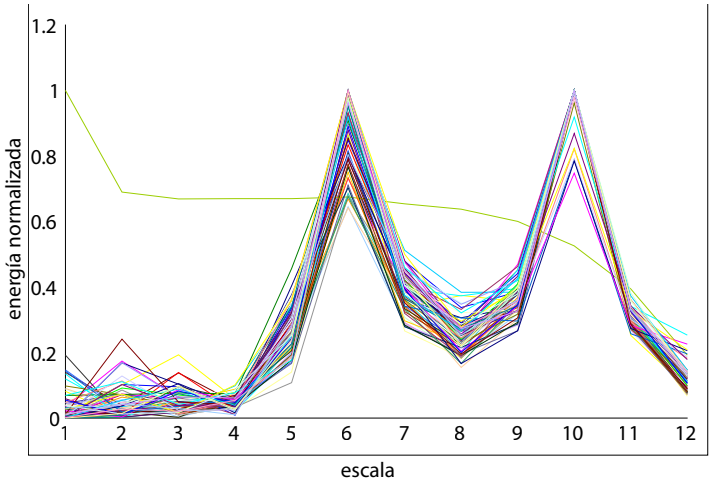
(m) DWT nivel 6 para ambos compresores

(n) DWT nivel 6 para un compresor

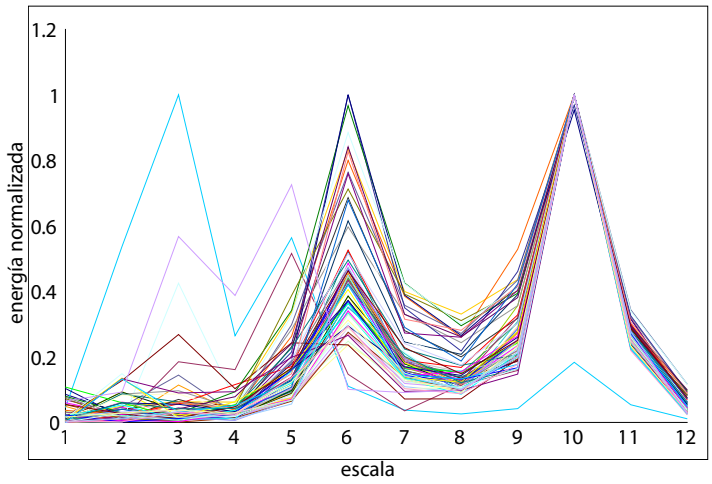
(o) DWT nivel 5 para ambos compresores

(p) DWT nivel 5 para un compresor

Figura 6.7: Gráficos de la DWT para ambas señales de prueba

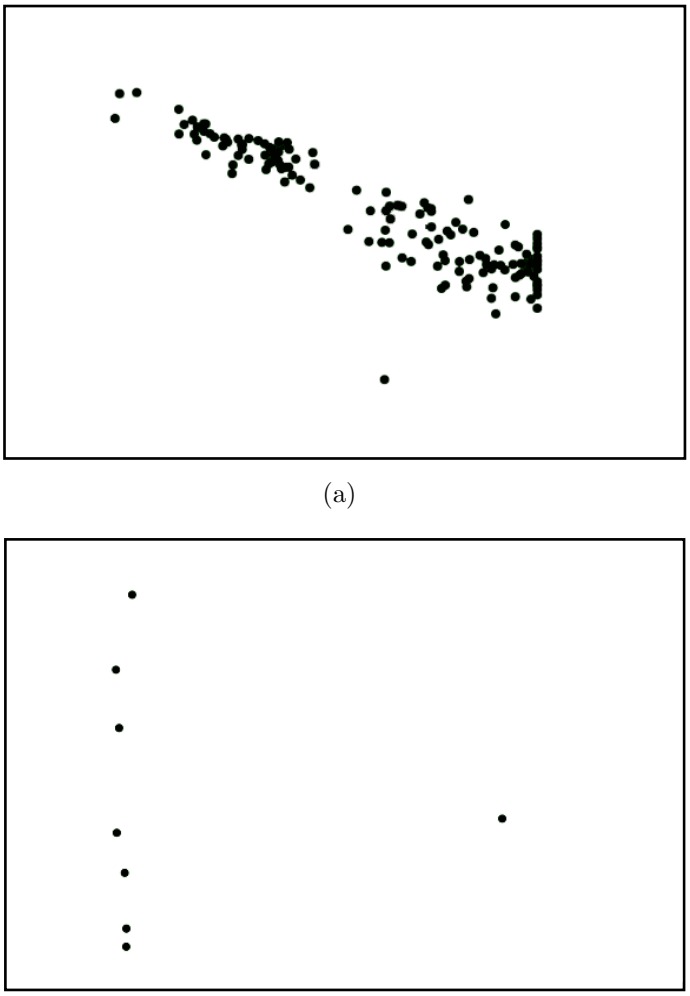


(a)



(b)

Figura 6.8: Vectores de características (a) para ambos compresores en funcionamiento y (b) para un solo compresor en funcionamiento



(a)

(b)

Figura 6.9: Gráficos de Sammon (a) para los datos de entrenamiento tomados para ambos compresores funcionando y un solo compresor funcionando y (b) para el codebook resultante del entrenamiento

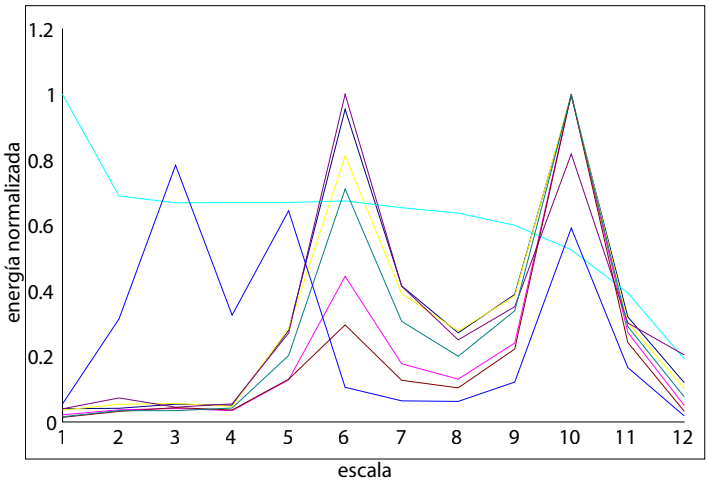


Figura 6.10: Codebook resultante del entrenamiento con las señales de vibración de los compresores.

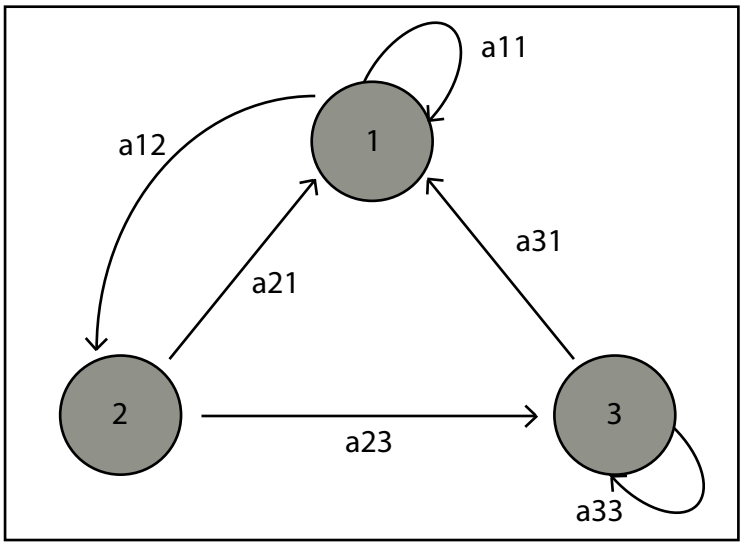


Figura 6.11: Esquema del Modelo de Markov resultante para la clasificación de las señales.

Tabla 6.1: Probabilidades resultantes de la aplicación de los Modelos de Markov calculados para el experimento con uno y dos compresores funcionando

Muestra	Single	dual
1	-17.533206933163569	1.0
2	-34.065352062009936	-9.1658346786744147
3	-12.383530196021708	1.0
4	-12.420059881876966	1.0
5	1.0	1.0
6	1.0	-16.269086157348383
7	1.0	1.0
8	1.0	1.0
9	1.0	-9.7606526354521712
10	-27.022273268260157	-9.428437296843283
11	-9.1105148558612878	1.0
12	1.0	-9.3802639228436053
13	1.0	1.0
14	1.0	-28.393917222705575
15	1.0	1.0

6.2 Análisis de resultados

Para el análisis de los resultados se dividirán los datos de acuerdo a como se ha hecho en todo el documento, tomando primero la solución referente a la herramienta para la generación de reportes para LabMonitor y luego para la herramienta de clasificación de condición.

Como se mencionó en la sección 5.1.1, para la herramienta para la generación de reportes se optó para la configuración de la herramienta, una base de datos en donde se almacenan los mensajes de texto para los reportes, así como las posiciones de memoria que son de interes dentro del PLC, y finalmente la configuración de red necesaria para la conexión con el servidor. En las figuras 6.3, 6.2 y 6.2, se muestran capturas de pantalla de la interfaz desarrollada para la configuración de la herramienta.

Para demostrar el éxito de la configuración, se muestra en la figuras 6.4 y 6.5, un grupo de capturas de paquetes de red, correspondientes a una sesión realizada utilizando la configuración mostrada en las primeras capturas de pantalla. Estas capturas de paquetes fueron realizadas utilizando el analizador de protocolos de código abierto Ethereal, en estas capturas se obtiene un desgloce de la estructura de un paquete de EthernetIP, puede observarse las 4 capas del modelo TCP/IP descrito en secciones anteriores; tal como se puede ver en las figuras mostradas, la estructura construida por medio de la librería EthernetIP desarrollada en la solución del problema es debidamente identificada por Ethereal como un paquete perteneciente a EthernetIP, esto permite deducir que la de primera mano que la estructura propuesta por la librería cumple con las reglas impuestas por el protocolo y es de esperarse como se vera en breve el éxito de la comunicación con el dispositivo remoto.

El primer caso corresponde al registro de la sesión entre el servidor y el cliente remoto, de acuerdo a lo explicado en la sección 3.2.2, al iniciar la comunicación entre el cliente y el servidor, el primero solicita el registro de la sesión en el segundo dispositivo (figura 6.5(a)), como respuesta a la solicitud el servidor envía la respuesta con un valor incluido correspondiente a un manejador de sesión (figura 6.5(b)), el cual será utilizado para envío por parte del cliente de los siguientes comandos.

En las siguientes capturas de pantalla se muestra el comando para la lectura de la memoria del PLC, como se muestra en la figura 6.6(a), el valor del manejador de sesión es retenido del comando de registro de sesión descrito anteriormente, y como se muestra en la estructura del comando enviado se cumplen con las especificaciones necesarias para la lectura de la memoria. En la figura se muestra entonces el resultado de la consulta, en donde finalmente los datos solicitados son devueltos por parte del PLC.

Para el análisis de la herramienta de clasificación de condición, el experimento realizado

consistió en la simulación de un problema en una de las cámaras de choque térmico, en donde el equipo fue montado sobre el compresor dedicando al control del sistema de refrigeración superior de la cámara (cada una de las cámaras cuenta con dos sistemas de refrigeración en cascada, en donde el sistema superior se encarga de enfríar al inferior el cual a su vez se encarga de enfríar las cámaras durante su funcionamiento). Para adherir el sensor de vibración en el compresor se agrego una placa sobre la cual se montó el sensor y se utilizó una cinta de doble cara para sujetarlo al compresor (aunque esto no parezca realmente importante si lo es). Se tomaron 1000 muestras, cada una de 4096 muestras en total, lo que correspondió a aproximadamente 15 minutos de muestreo continuo (lo que también es importante aunque no lo parezca). Durante este periodo se forzó al compresor del sistema de refrigeración inferior a detenerse, de forma que se obtendría muestras de ambos compresores funcionando y de un solo compresor funcionando, siempre claro está tomadas desde el punto de vista del compresor del sistema superior. Hay que hacer notar que la detención de uno de los compresores provoca que la cámara automáticamente apague el compresor que queda funcionando en solitario, esto porque las cámaras no pueden ser utilizadas con solo un compresor funcionando evitanto el colapso del sistema, es por esto que la cantidad de muestras con un solo compresor en funcionamiento es una parte reducida del todal de muestras, obteniendo además algunos estados de silencio que no son importantes para la experimentación.

En las figuras 6.7(a) y 6.7(b) se muestran dos de los 1000 paquetes de 4096 muestras obtenidos durante el experimento. En el primer caso se trata de ambos compresores trabajando y en el otro solamente se tiene uno en funcionamiento. En las señales se observa el resultado obtenido por el hardware de adquisición de datos desarrollado, el ancho de banda de estas señales es de aproximadamente 2500Hz de acuerdo con el dato del fabricante (la experimentación con el sensor de vibración mostró que el ancho de banda del acelerómetro se vió aumentado con el cambio de capacitor efectuado), dado que el muestreo se realizo a una velocidad mayor que esta no se tienen problemas de "aliasing" en las señales. Como se observa en las figuras no se tiene una distinción visual de las diferencias entre estas, pero el análisis en frecuencia realizado demostró estas diferencias.

En la figura 6.7 se muestra la DWT para las señales de las figuras 6.7(a) y 6.7(b), en esta se tienen las primeras 8 escalas, el resto no fue tomado en cuenta debido a que presentan características triviales para el análisis. Estos gráficos son el resultado de la aplicación de la clase "DSPHandler" descrita en la sección. En estas figuras aún no se muestra una clara diferencia entre las dos condiciones analizadas, sin embargo se nota que en el caso de las figuras 6.8(n) y 6.8(m), las cuales corresponden a las DWT de nivel 6 de cada señal se tiene una disminución en el caso de la señal para un solo compresor en funcionamiento. De forma que se tiene que en la señal correspondiente a un solo compresor se tiene una perdida en la aparición de una de las bandas de frecuencias mostradas por la DWT, la

cual será mas visible en breve.

En las figuras 6.9(b) y 6.9(a) se muestran los vectores de características resultantes del análisis con la clase "DSPHandler" a partír de las bandas de frecuencias de dos grupos compuestos de 70 paquetes de muestras seleccionados de entre los 1000 grupos disponibles que fueron señalados como pertenecientes al grupo de muestras en ambos casos. En los dos casos se muestran los 12 niveles de la DWT resultante. En el primer caso se muestran los vectores de características para ambos compresores en funcionamiento. Es extremadamente apreciable las características mostradas por los 70 paquetes de muestras, a excepción de uno que puede ser tomado como un ruido en el muestreo. Es obvio como se observa el funcionamiento de ambos compresores, en donde cada uno de los dos picos corresponden a uno de los compresores, de forma que la vibración de estos se encuentra "montada" principalmente en una banda de frecuencias separadas una de la otra. Es de notar además que las bandas de frecuencia de 1 a 4 no son de importancia y pueden ser no tomadas en cuenta para el análisis ya que el patrón se encuentra mas bien en los niveles superiores, pero para el caso de este experimento si fueron tomados en cuenta para mostrar el efecto del ruido sobre el reconocimiento, inclusive la muestra que se encuentra muy separada de las demás fue tomada en cuenta con el mismo propósito.

En la siguiente figura (6.9(a)), se muestra el análisis para el caso de uno de los compresores en funcionamiento. Como se observa en estas muestras se tiene una clara tendencia del pico correspondiente al nivel 6 de transformación a disminuir su amplitud de forma que se separa de las muestras para ambos compresores en funcionamiento. El comportamiento de esta gráfica se muestra degenerativo para la amplitud del segundo pico de frecuencia, debido a que las muestras tomadas para esta condición fueron tomadas en un rango a partir de donde desaparece la señal por la ausencia de ambos compresores, de esta forma la cantidad de muestras que muestran el comportamiento con un solo compresor es algo variable y por esto se muestra esta degradación. Como en el caso de las primeras muestras se tienen algunos puntos de ruido, correspondientes especialmente a los primeros 4 niveles, pero de nuevo estos fueron tomados en cuenta para ver su efecto en los resultados (como se verá más adelante eliminando estos 4 niveles del análisis se mejora en mucho el reconocimiento).

El siguiente paso en el proceso de reconomiento de la señal de vibración requiere obtener el codebook correspondiente a las muestras de entrenamiento que reflejen los estados de interés. En la figura 6.10 se muestra el codebook resultante al aplicar la clase "CodebookHandler" a los vectores de entrenamiento. Como se observa en la gráfica los vectores miembros del codebook son una muestra muy representativa de todo el conjunto de entrenamiento utilizado (que corresponden a las figuras 6.9(b) y 6.9(a)), como se observa se tienen ubicados inclusive aquellos vectores que corresponden al ruido introducido en el muestreo, así como las muestras que corresponden a ambos compresores en funciona-

miento como para solo un compresor en funcionamiento.

Como ayuda para la visualización de los datos y como se distribuyen estos unos de otros se muestra en la figura 6.10(a) el Mapa de Sammon para los datos de entrenamiento utilizados. Esto corresponde, como se explicó en la sección 3.2.7, a una vista en dos dimensiones de los vectores de características de 12 dimensiones utilizados durante el entrenamiento. Como se observa en esta figura, se muestra claramente dos grupos de datos. Los que corresponden a la parte inferior izquierda, corresponden a los datos con solo un compresor en funcionamiento esto debido a que se muestra la degradación desde la parte superior hasta la casi desaparición de estos puntos, los puntos en la mitad del gráfico corresponden a los datos para ambos compresores en funcionamiento, donde se observan mas agrupados y con una tendencia más clara. En la esquina superior izquierda, se muestran algunos puntos que corresponden a ruido de las muestras, además se observa un punto adicional en la parte inferior que también corresponde a un punto de ruido.

En la figura 6.10(b) se muestra el Mapa de Sammon para el codebook resultante, en esta figura se tienen los ocho puntos del codebook, la orientación de estos puntos varia con respecto a los del mapa de los datos de entrenamiento debido a propiedades del algoritmo de los Mapas de Sammon, pero en este se observa una tendencia similar a los datos de entrenamiento. Primero se observa un punto en solitario hacia la derecha que corresponde a un punto de ruido en las muestras, se puede observar a demas como se dan tres agrupaciones de datos hacia la izquiera, uno en la parte inferior, uno en el medio y otro en la parte superior, este ultimo corresponde a los otros valores de ruido en las muestras.

Una vez demostrada la efectividad de la discretización de las muestras, lo que queda es observar los resultados obtenidos a partir de los Modelos de Markov entrenados a partir de las secuencias de entrenamiento.

En la figura 6.11 se muestra un esquema del Modelo de Markov propuesto para la experimentación, el cual debido a la efectividad en el reconocimiento no fue variado para ninguna de las pruebas realizadas, cabe indicar que uno de los puntos de variación que podrían ser tomados en cuenta para aumentar la efectividad de la herramienta es la proponer otros tipos de modelos y ver cual es el que maximiza los resultados de reconocimiento o bien si no existe variación alguna. El modelo resultante posee tres estados, en donde la matriz de probabilidad inicial π fue propuesta para que el modelo comenzara siempre por el estado 1 y de acuerdo al codebook desarrollado se tiene un alfabeto de 8 miembros. En las siguientes expresiones se muestran las condiciones iniciales para el modelo propuesto

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & 0 & a_{23} \\ a_{31} & 0 & a_{43} \end{bmatrix} \tag{6.1}$$

$$B = \begin{bmatrix} 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\ 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \\ 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 & 1/8 \end{bmatrix} \tag{6.2}$$

$$\pi = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{6.3}$$

En la tabla 6.1 se muestran algunos resultados obtenidos a partir de la evaluación de muestras aleatorias contra los dos modelos cálculados a partir del estado anormal y normal de los compresores, como se explicó en la sección 5.3.4, los valores obtenidos a partir de la evaluación de una secuencia contra alguno de los modelos creados es igual al logaritmo de la probabilidad obtenida según la librería, por eso los valores obtenidos son valores negativos decrecientes hasta llegar a un valor de 0, puesto que toda probabilidad debe de sumar 1. Como se muestra en la tabla se obtuvieron valores iguales a 1 para los casos en los cuales los modelos desconocieron al 100% las muestras analizadas, esto es no pudieron saber a que fuente pertenecían estas muestras. Como se muestra en el conjunto de observaciones se dio un reconocimiento bastante claro de cuales muestras pertenecen a cada una de las fuentes analizadas, existen algunos casos como en el caso de las muestras 2 y 10 en los cuales se daba alguna probabilidad de que ambos modelos hubieran generado las muestras, esto es debido fundamentalmente al comportamiento de los vectores de características utilizados en el entrenamiento, donde algunos de estos eran bastantes similares en una y otra fuente.

El exito obtenido por la herramienta es de aproximadamente a un 67% tomando en cuenta todas las bandas de frecuencias de la señal analizada, así como los vectores de características que corresponden a valores de ruido. Es por esto que se requerirá entonces tener una gran variabilidad a la hora de tomar las muestras para el entrenamiento, según los autores se recomienda tomar aproximadamente un total de 1000 veces el tamaño del codebook, para asegurarse de que se captará toda la variabilidad posible de la señal, una vez realizado esto se debe de analizar las muestras para visualizar la variabilidad contenida en las distintas muestras de entrenamiento y entrenar los modelos según se requiera. Como se observa el proceso de entrenamiento puede ser un proceso que requiere de algo de experimentación pero todo lleva a aumentar los porcentajes de reconocimiento de la herramienta.

El presente analisis muestra entonces claramente el cumplimiento de los objetivos propuestos y además de la meta propuesta, dado que se realizó la implementación de los componentes necesarios para alcanzar la solución propuesta y en un alto porcentaje se alcanzó la resolución del problema propuesto.

Capítulo 7

Conclusiones y recomendaciones

7.1 Conclusiones

1. La comunicación con los PLC de las cámaras de choque térmico requiere del uso de la interfaz 1761 NET-ENI y el uso del protocolo EthernetIP.

2. El uso de la DWT permite obtener las características de señales de vibración que las distinguen en el dominio de la frecuencia.

3. El uso del algoritmo LBG para el entrenamiento del codebook permite recopilar el máximo de variabilidad de las muestras analizadas dentro de los miembros del codebook.

4. Los modelos de markov permiten la clasificación de secuencias discretas a través del entrenamiento de los modelos a partir de secuencias de entrenamiento conocidas.

5. El nivel de ruido en las muestras aplicadas a los Modelos de Markov entrenados afecta significativamente el porcentaje de reconocimiento de la herramienta de clasificación de condición.

6. El porcentaje de reconocimiento para la herramienta de clasificación de condición varió entre el 67.4%.

7. Es necesario poder obtener la mayor cantidad de muestras posibles durante el entrenamiento para poder atrapar el máximo de variabilidad posible.

7.2 Recomendaciones para la empresa

1. Es necesario continuar con el desarrollo de la librería EthernetIP con el fin de contar con una herramienta de comunicación que permita, no solo la extracción de información de los equipos que utilizan este protocolo sino también el control remoto de los equipos a través de redes TCP/IP.

2. Para la herramienta de clasificación de condición con el fin de obtener una herramienta más robusta es necesario primero encontrar un algoritmo de entrenamiento para el codebook que sea más efectiva que el algoritmo LBG, debido a que para el reentrenamiento del codebook es necesario destruir el codebook y reconstruirlo de nuevo, existen muchas otras alternativas para esta parte de la herramienta.

3. Es necesario ampliar el uso de herramientas dereconocimiento de patrones e inteligencia artificial en los equipos de laboratorio, por lo que se recomienda el desarrollo de herramientas basadas en redes neuronales y lógica difuza las cuales son ampliamente utilizadas en este tipo de aplicaciones.

Bibliografía

[1] Data compression principles [online]. Noviembre 2003 [visitado el 24 de noviembre de 2006]. URL http://www.data-compression.com/vq.shtml.

[2] Allen-Bradley. *DF1 Protocol and Command Set Reference Manual*. Allen-Bradley, Milwaukee WI, Octuber 1996.

[3] Allen-Bradley. *panelbuilder32 PanelBuilder32 User Manual*. Allen-Bradley, Milwaukee WI, Marzo 2002.

[4] Allen-Bradley. *panelbuilder32manual PanelBuilder32*. Allen-Bradley, Milwaukee WI, Marzo 2002.

[5] Allen-Bradley. *1747-rm001 SLC 500 Instruction Set*. Allen-Bradley, Milwaukee WI, Noviembre 2003.

[6] Allen-Bradley. *1761-pp004 Product Profile 1761-NET-ENI/w*. Allen-Bradley, Milwaukee WI, Octubre 2005.

[7] Allen-Bradley. *enet-br001 Ethernet/IP flyier*. Allen-Bradley, Milwaukee WI, Mayo 2005.

[8] Historic Naval Ships Association. About environmental chambers [online]. Octubre 2004 [visitado el 24 de noviembre de 2006]. URL http://www.hnsa.org/doc/fleetsub/refrig/index.htm.

[9] Peterson Bruce. *Enviromental Screen Screenning Tutorial*. Accolate Engineering Solutions, Irvine CA, 1998.

[10] Zilog Corporation. *PS0192 ez80f91 MCU*. Zilog Corporation, San Jose CA, Abril 2004.

[11] Ethereal. Ethereal network protocol analyzer [online]. Noviembre 2006 [visitado el 24 de noviembre de 2006]. URL http://www.ethereal.com/.

[12] Free Software Foundation. Gnu scientific library [online]. Noviembre 2006 [visitado el 24 de noviembre de 2006]. URL www.gnu.org/software/gsl/.

[13] MPI Molecular Genetics. Ghmm hidden markov library [online]. Febrero 2006 [visitado el 24 de noviembre de 2006]. URL http://www.ghmm.org.

[14] DiBartolomeo Gino. *Enviromental Testing Expendable Refrigerants or Mechanical Refrigeration*. Enviromental Stress Systems, Sonora CA, 2002.

[15] Anders Hedström. C++ sockets library [online]. junio 2000 [visitado el 24 de noviembre de 2006]. URL http://www.alhem.net/Sockets/.

[16] National Instruments. *AN007 Data Acquisition Fundamentals*. National Instruments, Milwaukee WI, Abril 2002.

[17] Texas Instruments. *Accelerometers and how they work*. Texas Instruments.

[18] Wang Litao, Mehrabi Mostafa, G., and Jr Elijah, Kannatey-Asibu. *Tool wear monitoring in reconfigurable machining systems through wavelet analysis*. Engineering Research Center for Reconfigurable Machining systems University of Michigan, Michigan.

[19] Robi Polikar. The wavelet tutorial [online]. Noviembre 1994 [visitado el 24 de noviembre de 2006]. URL http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html.

[20] Eddy Sean, R. *What is a Hidden Markov Model?*, volume 22. Nature Biotechnology, 10 edition, Octubre 2004.

[21] Cincinnati Sub-Zero. *Enviromental Stress Screenning*. Cincinnati Sub-Zero, Cincinatti OH, 2001.

[22] Swig. Swig wrapper [online]. mayo 2006 [visitado el 24 de noviembre de 2006]. URL http://www.swig.org.

[23] Lakshminarayanan V. *Envorimental Stress Screnning improves electronic-design reliability*, volume II. EDN magazine, 3 edition, Junio 2001.

[24] Petrushin Valery, A. *Hidden Markov Models: fundamentals and applications*. Online Symposium for Electronics Engineer, Chicago Il, 2000.

[25] Wikimedia. Wikipedia [online]. Diciembre 2004 [visitado el 24 de noviembre de 2006]. URL http://en.wikipedia.org/wiki.

[26] Achim Gaedke y Pierre Schnizer. Pygsl: Python interface for gnu scientific library [online]. Octubre 2003 [visitado el 24 de noviembre de 2006]. URL http://pygsl.sourceforge.net/.

Apéndice A

Notación

1. $p = P(A|B)$: Probabilidad condicional de A dado B.
2. R^k : Conjunto de vectores de dimensión k.
3. $\| X \|$: La norma del vector X donde $X \in R^k$, así $X = x_1, x_2, \cdots, x_k$, de forma que la norma de x está dada por

$$\| X \| = \sqrt{x_1^2 + x_2^2 + \cdots + x_k^2} \tag{A.1}$$

4. $\bigcup V_i$: La unión de los conjuntos de vectores V_i.
5. $\bigcap V_i$: La intersección de los conjuntos de vectores V_i.

Apéndice B

Glosario

bps Bits per second, bits por segundo, medida utilizada para contabilizar el ancho de banda digital de un dispositivo.

Chipset Circuito de aplicación dedicado a dar soporte a las funciones del microprocesador controlando funciones como acceso a memoria y control de periféricos.

Clustering Técnica de análisis de datos que utiliza particionamiento de datos en subconjuntos que comparten características comunes.

Codebook En criptografía, un codebook es un documento usado para implementar un código. Un codebook contiene una tabla de búsqueda para codificación y decodificación; cada palabra o frase tiene (uno o más) miembros del codebook. Para descifrar los mensajes cada final debe tener disponible una copia del codebook.

Codeword Palabra en clave.

Coeficiente wavelet Representación en el dominio del tiempo y la frecuencia para un conjunto discreto.

Conjunto Totalidad de los entes matemáticos que tienen una propiedad común.

Compresor En un sistema de refrigeración dispositivo encargado de aumentar la presión del refrigerante en forma de vapor que viene del evaporador y pasa al condensador.

DF1 Protocolo de redes industriales propietario de Allen-Bradley, basado en RS-232, permite hasta 255 nodos conectados y tiene tres capas implementadas: capa física, capa de enlace de datos y capa de aplicación.

Estocástico Teoría estadística de los procesos cuya evolución en el tiempo es aleatoria, tal como la secuencia de las tiradas de un dado.

HMM Hidden Markov Model, esto es Modelo Oculto de Markov.

Microprocesador Circuito constituido por millares de transistores integrados en un chip, que realiza alguna determinada función de los computadores electrónicos digitales.

OSI Open Systems Interconnection Reference Model, modelo de referencia de sistemas abierto, descripción por capaz de las comunicaciones y el diseño de redes de computadoras.

PLC Programmable logic controller, controlador lógico programable

Probabilidad En un proceso aleatorio, razón entre el número de casos favorables y el número de casos posibles.

Q&R Quality and Reliability departament, esto es Calidad y confiabilidad departamento.

MySQL Sistema de manejo de bases de datos tipo SQL con características de multiproceso y multiusuario. .

Señal Función de una o más variables con contenido semántico para alguna aplicación.

Señal estacionaria Señal cuyas componentes de frecuencia son constantes a través del tiempo contraria a una señal no estacionaria.

Vibración Cada movimiento vibratorio, o doble oscilación de las moléculas o del cuerpo vibrante.

Apéndice C

Descripción de la empresa

Componentes Intel de Costa Rica comenzó sus operaciones en marzo de 1998 con sus dos plantas de manufactura y centro de distribución en La Ribera de Belén, Heredia. Con una inversión acumulada de 420 millones, estas instalaciones conforman las únicas instalaciones de Intel en Latinoamérica y unos 2200 empleados directos de Intel brindan sus servicios.

La actividad principal de Componentes Intel de Costa Rica se resume en ensamble y prueba de microprocesadores y chipsets manufacturados por Intel a nivel mundial. Esta empresa ha sido líder en la aplicación de los más altos estándares y políticas de medio ambiente, salud y seguridad ocupacional, lo que ha beneficiado a otras empresas nacionales.