DISEÑO Y SIMULACIÓN DE UN DIAGNOSTICADOR DE FALLAS EN SENSORES Y ACTUADORES USANDO UN ENFOQUE DESCENTRALIZADO PARA UN SISTEMA DE BANDAS TRANSPORTADORAS DE PRODUCTOS

DISEÑO Y SIMULACIÓN DE UN DIAGNOSTICADOR DE FALLAS EN SENSORES Y ACTUADORES USANDO UN ENFOQUE DESCENTRALIZADO PARA UN SISTEMA DE BANDAS TRANSPORTADORAS DE PRODUCTOS

AUTOR:

FRANCISCO EDUARDO ALVARADO CORREA

Ingeniero Electrónico

Egresado de la Maestría en Control y Automatización Industriales

DIRIGIDO POR:

MARIELA CERRADA LOZADA, Ph.D.

Universidad de Los Andes, Mérida-Venezuela

Investigadora del Programa Prometeo

CUENCA – ECUADOR

2017

Datos de catalogación bibliográfica

ALVARADO CORREA FRANCISCO EDUARDO

DISEÑO Y SIMULACIÓN DE UN DIAGNOSTICADOR DE FALLAS EN LOS SENSORES Y ACTUADORES USANDO UN ENFOQUE DESCENTRALIZADO PARA UNA SISTEMA DE BANDAS TRANSPORTADORAS DE PRODUCTOS

Universidad Politécnica Salesiana, Cuenca – Ecuador, 2017

MAESTRÍA EN CONTROL Y AUTOMATIZACIÓN INDUSTRIALES

Formato 170 x 240 mm Páginas: 134

Breve reseña de los autores e información de contacto:

Autor:

FRANCISCO EDUARDO ALVARADO CORREA
Ingeniero Electrónico
Egresado de la Maestría en Control y Automatización Industriales
edualva01@gmail.com

Dirigido por:

MARIELA CERRADA LOZADA, Ph.D
Universidad de Los Andes, Mérida-Venezuela
Investigadora del Programa Prometeo
cerradam@ula.ve

IMPRESO EN ECUADOR – PRINTED IN ECUADOR.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

DEDICATORIA

El presente trabajo está dedicado a mis hijos, para que sirva de ejemplo y punto de partida en sus logros personales; y, de una forma especial a mi madre, quien siempre ha estado apoyándome incondicionalmente y alentándome a proseguir con mi preparación académica.

Francisco Eduardo Alvarado Correa

PREFACIO

El presente trabajo aplica una estrategia de diseño y la correspondiente simulación de un diagnosticador de fallas basado en eventos discretos, para sensores y actuadores usando un enfoque descentralizado para un sistema de bandas transportadoras de productos; el cual tiene como base un sistema de varios diagnosticadores locales que actuando de manera coordinada logran determinar la existencia una falla.

Para diseñar el diagnosticador descentralizado se partirá de las consideraciones de diseño de un diagnosticador monolítico modelado con Autómatas de Estado Finito (AEF), se definirán los diferentes módulos de los subsistemas existentes y se generarán los AEF de cada uno, considerando sus eventos observables, controlables, eventos de falla, así como los controladores locales de eventos discretos relacionados con la operación del proceso. Para cada subsistema se considera el diseño de un diagnosticador local, y la información de diagnóstico se obtiene a partir del análisis de los eventos observables provenientes de los sensores en un momento dado, usando un algoritmo de coordinación y un protocolo de comunicación entre los diagnosticadores locales.

Como caso de estudio se considerará el funcionamiento del motor trifásico que pone en movimiento un sistema de bandas transportadoras, que es actuado por un contactor electromecánico de potencia y tiene sensores dispuestos de la siguiente manera: dos sensores complementarios colocados en el contactor de potencia, y otro sensor para indicar el movimiento rotatorio del eje del motor.

Al final se presentará la operación mediante un simulador, el cual permitirá, de manera didáctica, verificar la presencia de alguna falla en cualquiera de las tres bandas específicas de demostración. Este simulador puede ser implementado en un SCADA, simplemente considerando los eventos, en tiempo real, generados por los sensores y el controlador.

PRÓLOGO

Para desarrollar una estrategia de diseño y simulación de un diagnosticador de fallas en sensores y actuadores usando un enfoque descentralizado para un sistema de bandas transportadoras de productos, se presentan cinco capítulos que tienen como objetivo exponer de manera sistemática y comprensible los planteamientos teóricos que permiten lograr el diseño del diagnosticador descentralizado usando un enfoque modular.

El desglose de estos capítulos es:

- Capítulo 1: Introducción, presenta el problema planteado, los antecedentes de solución, y los objetivos que conllevan a la contribución de este trabajo de investigación.

- Capítulo 2: Fundamentos Teóricos, aquí se expone los fundamentos teóricos generales necesarios para comprender el enfoque de modelado y solución del problema planteado. Se inicia con el formalismo teórico para entender el modelado de sistemas de eventos discretos a través de un autómata de estado finito. Se presentan algunos conceptos de varios autores para comprender de manera idónea, el planteamiento de un diagnosticador modular.

- Capítulo 3: Sistemas de Transporte y Distribución de Productos en Bandas Transportadoras, se expone el sistema industrial de aplicación. Este es un sistema de transporte de producto, ejemplo encontrado en todo tipo de industria de manufactura. Centrándose en las tres primeras bandas, ya que el resto del sistema es similar. También se establecen, por medio de la herramienta de software DESUMA V3.0, los autómatas requeridos, a partir de la definición de sus eventos observables y controlables, y de falla.

- Capítulo 4: Diseño de Diagnosticadores Descentralizados, se aplica la metodología para elaborar cada uno de los diagnosticadores modulares, por medio de un proceso secuencial de pasos. Se somete a discusión el desempeño de los diagnosticadores a través de cuatro ejemplos demostrativos para diferentes situaciones de falla de la banda 1. Hay que tener en cuenta que las otras dos bandas y sus

diagnosticadores, son la réplica exacta de la metodología aplicada a la banda 1.

- Capítulo 5: Ambiente de Simulación, aquí se presenta el funcionamiento del simulador, el cual se desarrolló en el lenguaje de programación “Java”. Se explica el proceso de trabajo del software y su operatividad demostrativa aplicada a los ejemplos de prueba.

AGRADECIMIENTO

Agradezco a mi madre por su constante aliento a proseguir mis estudios; a mis hijos que son mi motivo para seguir adelante; a la Psic. Clín. Cristina Arias por su impulso a desarrollar este trabajo; a los directivos de Post-Grado por su apoyo; y, en especial a la Ing. Mariela Cerrada, Ph.D., quien con paciencia y tiempo supo guiarme en todo el proceso de este trabajo.

Francisco Eduardo Alvarado Correa

CAPÍTULO 1

INTRODUCCIÓN

1.1 Definición del problema

En la mayoría de los procesos industriales se pueden encontrar procesos de diferente naturaleza, unos son procesos contínuos, que se pueden modelar y tratar con sistemas de ecuaciones diferenciales, de las cuales se logran obtener leyes de control con una exactitud adecuada, otros son modelados como Sistemas de Eventos Discretos (SED), consiguiéndose otra perspectiva de control, que requiere aplicar nuevas técnicas de control y supervisión. Un formalismo ampliamente conocido para el modelado y diseño de los sistemas de supervisión en SED es el uso de Autómatas de Estado Finito.

Desde el punto de vista de la supervisión, estos formalismos matemáticos establecen modelos que permiten observar funcionamientos del proceso, con el fin de visualizar las posibles anomalías que se presenten, identificándolas para posteriormente corregirlas y alcanzar un estado de recuperación de falla, asegurando estándares de operatividad que maximicen la producción industrial.

La detección de fallas en los actuadores y sensores de un sistema de bandas transportadoras, que específicamente se requieren para el cierre o apertura de compuertas, parada o movimiento de las bandas, se modelan como sistemas de eventos discretos a través de Autómatas de Estado Finito, a partir de los cuales puedan diseñarse unos diagnosticadores de fallas, lo que puede resultar altamente complejo por el crecimiento combinatorio de todos los posibles estados discretos del sistema. Es por ello, que el uso de enfoques descentralizados resulta de especial importancia para tratar la complejidad inherente a este tipo de sistemas. Por lo que el diseño de diagnosticadores de fallas menos complejos, para este tipo de sistemas, es un tópico de interés investigativo y aplicativo en entornos industriales reales, donde la implementación de diagnosticadores centralizados es un problema computacional que debe abordarse con contribuciones como la presentada en este trabajo.

Específicamente, se usarán autómatas de estado finito como formalismo de modelado, y enfoques modulares para tratar la descentralización. Con ello se espera aportar con nuevas ideas y argumentos a la practicidad de los conocimientos que puedan generarse sobre el diseño de diagnosticadores en esta clase de sistemas en la industria de manufactura, para el diseño de aplicación de supervisor con bajo coste computacional de implementación.

1.2 Antecedentes sobre el diagnóstico de fallas en sistemas industriales a base de eventos discretos

(Sayed-Mouchaweh, 2014) indica que una falla es una desviación del funcionamiento normal, reduciendo la operatividad de un sistema, o su capacidad de alcanzar funciones requeridas, siendo las posibles causas fallas por el uso de los componentes del sistema, fallas por malas condiciones de operación como sobrecarga, fallas de mantenimiento, fallas humanas, entre otros.

(Blanke, 1996) manifiesta que se puede entender que las fallas son hechos ineludibles en todo proceso, por lo que, con la finalidad de evitar deterioros en las máquinas y personas, estas fallas se deben hallar de manera rápida, además de tomar acciones para detener sus efectos.

Para esto, en el caso de los procesos modelados con Sistemas de Eventos Discretos (SED), se tiene, para diagnosticar una falla, dos tipos de diagnosticadores: los diagnosticadores monolíticos, y los diagnosticadores descentralizados.

Lo que nos lleva a plantear una pregunta evidente:

¿Es posible diseñar un algoritmo de diagnóstico descentralizado para verificar aquellas anomalías que interrumpirían el proceso productivo, que produzca la información diagnóstica de la misma manera que un diagnosticador centralizado?

Para responderla, es necesario revisar los avances recientes en torno a la proposición de enfoques distribuidos o modulares para diagnóstico de falla en sistemas de eventos discretos, que se discute a continuación.

Los diagnosticadores monolíticos son llamados también centralizados, porque todas las fallas son analizadas por él, una a la vez. (Sampath, 1995). Este implementa un modelo de la planta con los sensores y con los actuadores, y, se elaborara un alfabeto común, de donde se establecen los eventos observables y no-observables, controlables y no controlables debido al funcionamiento conjunto de los actuadores y sensores. Los eventos de falla se los asume que son eventos no-observables y no controlables, y se establece el controlador, con los eventos observables y controlables. Entonces, cuando se hace la combinación síncrona del modelo de la planta y del controlador, se obtiene el modelo del sistema controlado, a partir del cual se puede generar el modelo de un autómata diagnosticador, donde las dimensiones de tales modelos resultan intratables desde el punto de vista computacional para su implementación y para un análisis en tiempo real, particularmente cuando se trata de detectar y diagnosticar fallas.

Los diagnosticadores descentralizados, como por ejemplo los modulares, resuelven en parte este problema, ya que el diagnosticador global se expande en otros locales considerando algunos de los dispositivos involucrados.

Lo previamente expresado llevó a algunos trabajos de investigación que plantean soluciones sobre cómo se debería implementar este tipo de diagnosticadores; desde el punto de vista académico – práctico, se tiene:

• (Ricker & Fabre, 2000) presenta un enfoque modular para el diseño de observadores y diagnosticadores para la detección de fallas en un sistema distribuido, asumiendo que el modelo es compuesto por distribuciones que interactúan – cooperan unos con otros en función de un conjunto de recursos comunes. El planteamiento lo realizan a nivel de alarmas, que son emitidas ante cualquier falla. Los sistemas analizados son no – temporizados, acorde a la acción de las alarmas.

• (Contant, Lafortune, & Teneketzis, 2006) presenta como elaborar diagnosticadores de eventos discretos, enfocando la noción de diagnosticadores modulares en sistemas dinámicos. Basa su aplicación en un nuevo algoritmo que se aprovecha de forma incremental la estructura modular del sistema para ahorrar el esfuerzo computacional.

• (Zhou & et_al, 2008) aborda el diagnóstico de fallas por eventos discretos en sistemas descentralizados, con diagnosticadores locales que, en los sitios de los componentes, usan sus propias observaciones locales. Lo que asegura una escalabilidad respecto al número de módulos componentes, y lleva a que los diagnosticadores sean computables basados en los modelos locales, y no en el global. Entonces, no se pierden las detecciones, ya que cada falla es detectada en un número fijo de localizaciones, al igual que las falsas alarmas quedan inhibidas. La complejidad de la implementación de diagnosticadores locales se trata como polinómica en el número de módulos locales.

• (Gascar & Zineb, 2011) enfocan la detección de fallas de sistemas complejos por eventos discretos en el contexto de la comunicación temporizada de los autómatas por medio de canales. Desarrollan una metodología para detectar y aislar la falla, además, consideran que existen varios autómatas que están sincronizados en la comunicación, representando a los componentes del sistema. Por lo que se pretende implementar un algoritmo para calcular el diagnosticador, como también poder analizar la composición paralela de los modelos parciales, y así, aplicar un análisis de accesibilidad en la exploración del espacio de estados, dada por la prioridad más alta al controlador de transiciones.

• (Monsalve, 2011) en su trabajo de grado, hace un análisis al enfoque modular de los diagnosticadores, y lo compara con los diagnosticadores monolíticos. Basa su estudio en el trabajo: Modular fault diagnosis using temporized analysis for a class of discrete event system, (Cerrada, Ferrarini, & Dedé, 2010). Con esto plantea una

modificación al protocolo de coordinación para el diagnóstico modular, lo que significa cambios en las reglas de coordinación con las cuales se mezcla la información de los diagnosticadores locales. También propone cambios en la función de mezcla y se define una nueva función de emparejamiento para el protocolo de coordinación. La comunicación entre los diagnosticadores locales se basa en reglas de coordinación que usan la función de mezcla para estimar el estado global del sistema.

- (Ferrarini, Allevi, & A., 2011) identifica una falla de forma on – line, por un modelo temporizado y un algoritmo que se deriva del enfoque clásico de diagnosticadores, con el modelamiento explícito del tiempo de espera y del valor nominal y controlar el comportamiento no – nominal. Por lo que el objetivo de hallar una forma práctica y concreta de escribir un software que tenga la habilidad de aislar rápidamente y en línea una falla al dispararse una alarma, esto con enfoque en la automatización industrial. El método usado se basa en modelos de diagnóstico temporizados, entonces, el modelo opera en tiempo real manejando hardware industrial y ejecutando el software generado en un softPLC.

- (Ocheana, Popescu, & Ferrarini, 2013) presenta un método novel para la detección de fallas por eventos discretos, que se desarrolla en función del enfoque de un diagnosticador clásico. Consiste en reducir la cantidad de cómputo que se debe efectuar para aislar una falla lo más rápido posible. Esto se logra reduciendo el número de estados del modelo del proceso, llegando al modelo del sistema inclusive, con la consecuente reducción de los estados del diagnosticador. Para indicar esta conceptualización, se aplica al análisis de una estación de compresor. En concreto, se compara un diagnosticador clásico con el planteado.

- (González & Cerrada, 2014) emplea el uso de crónicas y análisis modular temporizado para proponer diagnosticadores modulares. Cada falla se asocia a un conjunto de crónicas, y cada crónica reconoce una firma de la falla, la cual es obtenida a partir de diagnosticadores de estado, modelados por autómatas de estado finito, para cada módulo definido. El conjunto de crónicas se manipulan por un protocolo de coordinación en línea, y se aplica en un caso de estudio sobre un sistema hidráulico.

- (Sayed-Mouchaweh, 2014) enfoca diferentes técnicas clasificadas por criterios de modelamiento (Autómatas, Redes de Petri). Al estudiar de forma práctica cada uno, se alcanza un método eficiente para el diagnóstico de fallas acorde a las restricciones de aplicación.

Al igual que la base teórica, que es escasa y puntual debido a lo relativamente nuevo de enfoque por eventos discretos; la práctica reportada también lo es. Pero aun así se logra conseguir aplicaciones específicas, tales como:

• (Morant, Quiles, Blasco, & Correcher, 2001) reporta la aplicación de diagnosticadores modulares basados en autómatas finitos, en un proceso de una columna de destilación. Además, este trabajo indica que los resultados fueron satisfactorios: produce una simplificación notable en el número de estados de los modelos resultantes, comparándola con la metodología original, aliviando de esta forma el problema de la complejidad computacional y pérdida de legibilidad de los modelos de salida y de los diagnosticadores resultantes. También, permite la aplicación de su metodología a sistemas de control extensos, complejos y distribuidos.

Adicionalmente, la aproximación modular, al tener diferentes diagnosticadores operando de forma concurrente, permite la localización de diferentes tipos de fallos que puedan producirse de forma local, incrementando, así, la capacidad de detección del sistema supervisor. Conjuntamente, la aplicación del concepto de controlador modular sincronizado, permite la aplicación de acciones de control de carácter múltiple sin pérdida del efecto acción/reacción en la respuesta del sistema.

Con base en toda esta información, el trabajo que se plantea serviría como base informativa real, dado que se aplicará a un proceso clásico industrial, obteniendo información valiosa que serviría de apoyo a trabajos de investigación futuros. También aporta a la industria a establecer métodos viables en el diagnóstico de fallas, resultando en una contestación, a priori, de la pregunta planteada:

Sí, efectivamente, sí sería posible la implementación de un sistema que colabore en el diagnóstico de fallas, de manera que ahorre esfuerzo computacional y posea un alto grado de confiabilidad, respecto al diagnóstico propuesto.

La investigación y desarrollo a realizar, se apoyará en los trabajos previos mencionados. Para hacer una propuesta de diseño modular de diagnosticadores de fallas aplicada a un proceso típico industrial, y emitir un aporte a la base de conocimiento técnico.

Como fundamento de partida teórico se considerará el libro de (Lafortune & Cassandras, 2008), el trabajo doctoral de (Sampath, 1995), así como el libro de (Sayed-Mouchaweh, 2014), y los trabajos de (Monsalve, 2011) y (Ocheana, Popescu, & Ferrarini, 2013).

1.3 Objetivos

1.3.1. Objetivo General

Diseñar un diagnosticador descentralizado de fallas en sistemas de eventos discretos, basado en autómatas de estado finito, para un sistema de transporte de producto a través de bandas transportadoras (TDP – BT).

1.3.2. Objetivos específicos

- Entender cómo opera un sistema de diagnosticador centralizado, y su diseño.
- Entender cómo opera un sistema de diagnosticadores descentralizado, y su diseño.
- Desarrollar el modelo controlado del sistema TDP-BT por eventos discretos.
- Diseñar un diagnosticador descentralizado para el sistema TDP-BT, basado en el modelo controlado del sistema.
- Desarrollar un ambiente de simulación para ilustrar el sistema de diagnóstico diseñado.

1.4 Justificación de la investigación

En los últimos tiempos se ha enfatizado en el análisis "Causa - Efecto" para dar una respuesta a los eventos de falla en sensores y actuadores que se ocasionan en una línea de producción, elaborándose técnicas muy variadas para poder analizar los eventos observados y detectarlas, antes de que conduzcan a algún desperfecto mayor en el sistema operativo productivo.

Entendiéndose como falla a aquella singularidad que lleva a un dispositivo a cambiar su estado de funcionamiento normal a anormal, su detección y diagnóstico es un campo aún abierto de investigación, que ahora dispone de resultados muy puntuales sobre el formalismo a emplearse, y muy poca información de la implementación sobre sistemas reales.

Al plantearse la realización de este trabajo de investigación, con la idea de estudiar y aplicar la teoría sobre diagnosticadores descentralizados (modulares o por grupos), a un sistema típico industrial, se llevaría a conclusiones que servirían para avanzar en la exploración de mecanismos que articulen de mejor manera la teoría orientada a la práctica. Siendo el análisis de esta aplicación válida tanto para la industria como para la academia.

CAPÍTULO 2

FUNDAMENTOS TEÓRICOS

2.1 Introducción

Antes de definir los modelos de Autómatas Finitos Deterministas y los No Deterministas, se hará un repaso de lo elemental de la teoría de Autómatas.

2.1.1 Alfabeto

Es un conjunto de símbolos, por lo general finito y no vacío. Un ejemplo de esto es:

El alfabeto castellano, que contiene 27 caracteres. Con los cuales se pueden hacer varias combinaciones.

(Kelley, 1995) indica que siendo Σ la letra que lo representa, entonces, σ ϵ Σ quiere decir que σ es un símbolo de Σ. Esto es:

$$\Sigma = \{a, e, i, o, u\}$$

Por lo que se puede decir que $a \epsilon \Sigma$.

2.1.2 Cadena (palabra = *w*)

Es un conjunto de caracteres finito que se enmarca en un alfabeto. Se define a una palabra dentro de un alfabeto, entonces si se tratara del alfabeto castellano, algunos ejemplos serían: sol, pradera, molecular y cerradura.

En consecuencia, una palabra se forma con cadenas de longitudes infinitas, a partir de un alfabeto. Además, se tiene la cadena vacía ε, que es aquella representación del conjunto vacío ∅. La longitud de una palabra $|w|$ es la cantidad de caracteres que la forman. (Kelley, 1995)

2.1.3 Lenguaje

Es el que se forma con varias cadenas, se representa por $L(\cdot)$. Pueden existir varios lenguajes en un mismo alfabeto, por lo que son de longitud infinita. Como también el lenguaje puede ser formal: conjunto de cadenas de símbolos tomados de algún alfabeto; y, otro tipo de lenguaje es el conjunto de cadenas sobre un alfabeto fijo. (Hopcroft & Ullman, 1993)

Por ejemplo, si $\Sigma = \{a\}$ se tiene:

$$\Sigma^* = \{\varepsilon\ , a\ , aa\ , aaa\ , \dots\}$$

donde Σ^* denota la cerradura de Kleene sobre un alfabeto dado.

2.1.4 Grafos

Con base en (Epp, 2011), un grafo es un conjunto no vacío de vértices y un conjunto no vacío de aristas, $G = (V, E)$, donde cada arista está asociada a un conjunto compuesto por uno o dos vértices llamados puntos extremos.

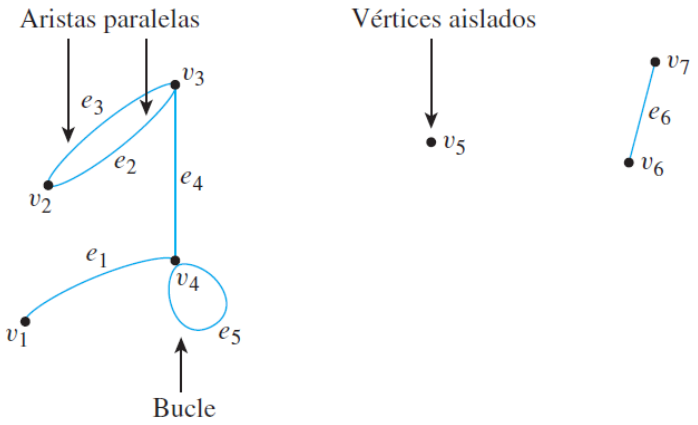


Figura 2. 1 *Representación de grafos y árboles.*
Fuente: (Epp, 2011)

La Figura 2. 1 indica cuáles son las aristas y los vértices, éstos están interactuando entre sí y forman el grafo respectivo que representa cierto conjunto de acciones tomadas o por tomarse.

Un grafo puede ser dirigido, ***digráfica***, cuando un conjunto finito no vacío de vértices y un conjunto de aristas dirigidas se asocian, cada uno, con un par ordenado de vértices denominados *puntos extremos*. (Ver Figura 2. 2)

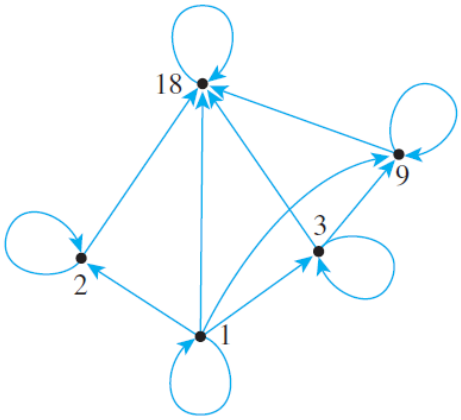


Figura 2. 2 *Diagrama de Hasse*
Fuente: (Epp, 2011)

2.2 Autómatas Finitos Deterministas y No Deterministas

2.2.1 Autómatas Finito Determinístico: AFD

Un autómata, en concreto, es una máquina secuencial capaz de manejar valores (palabras de entrada) en las entradas y producir otros valores en sus salidas (palabras de salida). Entonces, para entender cómo opera un AFD, se tomará el siguiente caso:

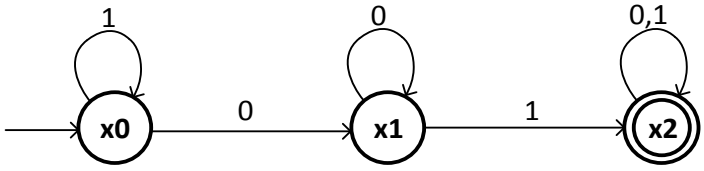


Figura 2. 3 *Digráfica que representa a un AFD*
Fuente: (Epp, 2011)

El grafo de la Figura 2. 3 indica que se tiene:

Estados: $X = \{x_1 , x_2 , x_3\}$

Estado inicial: x_0

Alfabeto: $\Sigma = \{0 , 1\}$

Estado de aceptación: $X_m = \{x_2\}$

Función de transición:

$\delta = (estado\ actual , elemento\ de\ entrada) = estado\ de\ llegada$, siendo para cada transición:

$$\delta = (x_0 , 1) = x_0 \qquad \delta = (x_0 , 0) = x_1 \qquad \delta = (x_1 , 0) = x_1$$
$$\delta = (x_1 , 1) = x_2 \qquad \delta = (x_2 , 0) = x_2 \qquad \delta = (x_2 , 1) = x_2$$

Se puede notar que cualquier cadena que inicie con el caracter '0', logra producir la transición de x_0 a x_1. Entonces, dentro del alfabeto se tiene que el lenguaje está formado por aquellas cadenas que inicien con este caracter, teniéndose un número infinito de combinaciones.

Pero las cadenas válidas o aceptadas serán aquellas que permitan llegar al estado aceptado. Estas pertenecerán al lenguaje que estará dentro del alfabeto. Por ejemplo:

La cadena 01000, es una cadena que permite la transición hasta el estado x_1, pero que no logra alcanzar el estado aceptado x_2. En cambio, la cadena 01100101 si es una cadena válida ya que permite el recorrido normal del autómata, logrando alcanzarse el estado aceptado x_2. Transformándose en una cadena del "lenguaje aceptado" → *"lenguaje regular".*

Por lo que dado estas características, se concluye que: *para cualquier valor de entrada existe un único estado que puede alcanzar el autómata*.

Pudiéndose definirse ya formalmente a este AFD como una 5-tupla, conformada por:

$$A = (X, \Sigma , \delta , x_0 , X_m)$$

En donde:

X = conjunto de estados.

Σ = conjunto de eventos finitos del autómata, en donde $\Sigma = \Sigma_o \cup \Sigma_{uo}$, y considerando que $\Sigma_c \in \Sigma_o$ y $\Sigma_f \in \Sigma_{uo}$, siendo Σ_c = eventos en el espacio

controlable, Σ_o = eventos en el espacio observable, Σ_f = eventos en el espacio de falla, Σ_{uo} = eventos en el espacio no observable.

δ = es la función de transición, siendo $\delta : X \times \Sigma \rightarrow X$.

x_0 = es el estado inicial.

X_m = corresponde al conjunto de estados marcados, siendo $X_m \subseteq X$.

Esta notación es importante para futuras referencias en este trabajo.

(Isasi, Martínez, & Borrajo, 1997) indican que un Autómata Finito (AF) genera un sólo tipo de salida aceptación, si están en un estado del conjunto X_m; o no aceptación si no están en un estado de X_m. El comportamiento es similar al de las Máquinas Secuenciales que comienzan en el estado inicial, y según se van recibiendo los símbolos en la entrada, estos transitan entre los estados, de acuerdo a la función de transición δ. Si en un determinado momento se encuentra en un estado de aceptación, reconoce como válida la palabra formada por los símbolos de entrada leídos hasta ese momento. Caso contrario no es aceptada.

Otro caso es el "de la Función de transición extendida", que partiendo de la definición formal del autómata, y haciendo un análisis del autómata de la Figura 2. 3, por la tabla de transición, es:

	Estados	Valores de transición	
*	x_x	0	1
→	x_0	x_1	x_0
	x_1	x_1	x_2
	x_2	x_2	x_2

Tabla 2. 1 *Tabla de transición de estados del AFD de la Figura 2.3*
Fuente: *Autor*

Mediante esta tabla se puede apreciar las secuencias que tiene el autómata, además, de observar las cadenas que acepta el lenguaje del autómata; siendo:

Cualquier cantidad de "1´s" en la entrada,

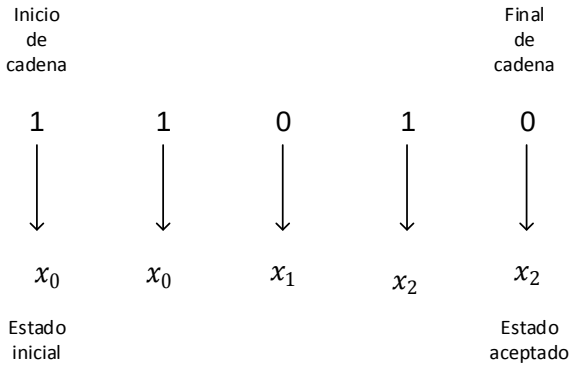
Se requiere un "0" para llegar a x_1, luego, cualquier cantidad de "0´s" en x_1,

Para llegar a x_2, luego de un "0", se demanda de un "1",

Al alcanzar x_2 , se puede tener cualquier otra combinación de "1´s" o "0´s".

Dada estas acciones, se ha logrado alcanzar el estado de aceptación x_2.

Si analizamos con la cadena "11010", se observa:



Dado que se logra alcanzar el estado aceptado x_2, y también se llega al final de la cadena, se considera esta cadena pertenece al *lenguaje* del autómata. Lo que la transforma en una *cadena aceptada*.

Esto nos lleva a una definición formal de la función de transición extendida:

$$\hat{\delta}(estado\ actual, \omega) = estado\ de\ llegada,$$

indica al estado que se llegaría previo procesamiento de la cadena ω. Por ejemplo:

$$\omega = 11010 \quad \Longrightarrow \quad \hat{\delta}(x_0, \omega) = x_2$$

En base a la definición de la función de transición extendida, se permite establecer el lenguaje del autómata finito determinista. Lo que define un lenguaje regular, dado de la siguiente manera:

$$L(A) = \{\omega | \hat{\delta}(x_0, \omega) = estado\ \epsilon\ X_m\},$$

entonces, este lenguaje regular es el aceptado por el AFD.

2.2.2 Autómata Finito No Determinístico: AFN

Al analizar el autómata de la Figura 2. 4, por medio del "Análisis de hilos", se tiene:

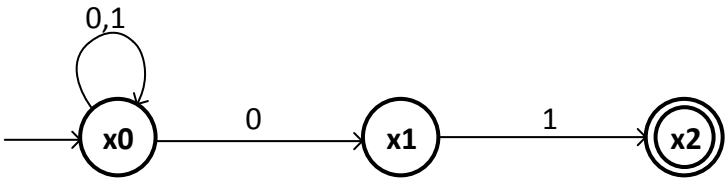


Figura 2. 4 *Digráfica que representa a un AFN.*
Fuente: (Epp, 2011)

Se le ingresa la palabra $\omega = 00101$, y al aplicar el análisis resulta la Figura 2. 5:

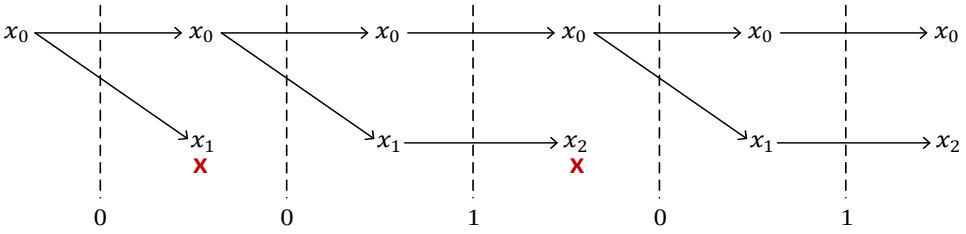


Figura 2. 5 *Gráfica de hilos que representa el recorrido de una palabra en un AFN*
Fuente: (Epp, 2011)

Leyendo la cadena de izquierda a derecha, se produce:

- Con el 1er caracter '0': se alcanza el estado x_0 y x_1.
- Con el 2do caracter '0': partiendo de los estados alcanzados anteriormente, se tiene:
  - Desde x_0 : se llega a sí mismo, y a x_1.
  - Desde x_1 : no se alcanza ningún otro estado, por lo que se descarta este ramal.
- Con el 3er caracter '1': partiendo de los estados alcanzados anteriormente, se tiene:
  - Desde x_0: se llega a sí mismo.
  - Desde x_1 : se llega a x_2.
- Con el 4to caracter '0': partiendo de los estados alcanzados anteriormente, se tiene:
  - Desde x_0 : se llega a sí mismo y a x_1.

- Desde x_2 : no se alcanza ningún otro estado, por lo que se descarta este ramal.

- Con el 5to caracter '1': partiendo de los estados alcanzados anteriormente, se tiene:
  - Desde x_0 : se llega a sí mismo.
  - Desde x_1 : se llega a x_2.

Entonces:

- Como se alcanza el estado x_2, se asume que la cadena que alcanza dicho estado es una *cadena aceptada*. Esto es, la cadena de lenguaje reconocido por el autómata.

Dado lo anterior, se tiene que se puede describir a un Autómata Finito no Determinístico como:

$$A = (X\ , \Sigma\ , \delta\ , x_0\ \ , X_m\ ),$$

que tiene la misma descripción que un determinístico, con la salvedad de que la función de transición tiene la forma:

$$\delta = (estado\ actual\ , elemento\ de\ entrada) = conjunto\ de\ estados\ de\ llegada,$$

por ejemplo:

$$\omega = 00101 \qquad \Longrightarrow \qquad \hat{\delta}(x_0, 0) = \{x_0, x_1\}$$

Por lo que se logra llegar a varios estados.

Con esta connotación, se razonará el siguiente autómata para definir la *función de transición extendida y el lenguaje* que maneja:

La función de transición está definida por $A = (X\ , \Sigma\ , \delta\ , x_0\ , X_m\ )$, con la cual se analizará el autómata de la Figura 2. 6, operando con la palabra $\omega = 0100$.

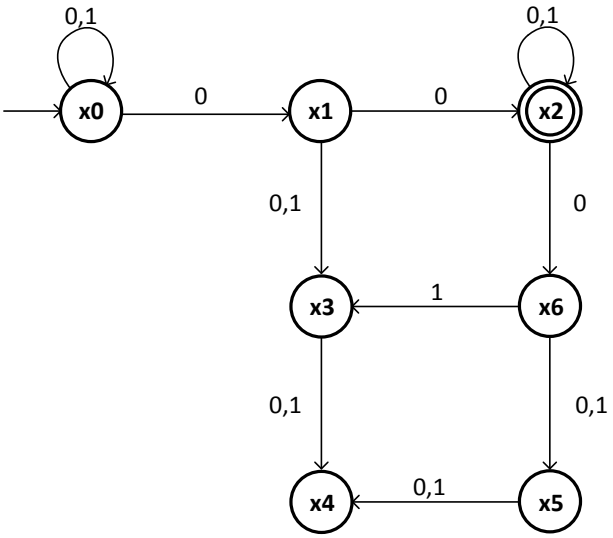


Figura 2. 6 *Gráfica que representa a un AFN*
Fuente: (Epp, 2011)

En donde se tiene que:

$$\hat{\delta}(x_0, 0) = \{x_0, x_1\} \qquad \hat{\delta}(x_0, 1) = \{x_0\} \qquad \hat{\delta}(x_1, 1) = \{x_3\} \qquad \hat{\delta}(x_0, 0) = \{x_0, x_1\}$$
$$\hat{\delta}(x_3, 0) = \{x_4\} \qquad \hat{\delta}(x_0, 0) = \{x_0, x_1\} \qquad \hat{\delta}(x_1, 0) = \{x_2, x_3\} \qquad \hat{\delta}(x_4, 0) = \{x_5\}$$

Con lo que la función de transición extendida queda definida por:

$$\omega = 0100 \qquad \Longrightarrow \qquad \hat{\delta}(x_0, \omega) = \{x_0, x_1, x_2, x_3, x_5\}$$

Sabiendo que $X_m = x_2$, se efectúa una operación de "intersección de conjuntos" entre la función de transición encontrada y los estados aceptados, siendo:

$$\{x_0, x_1, x_2, x_3, x_5\} \ \cap \ \{x_2\} \qquad \Longrightarrow \qquad \{x_2\},$$

lo que implica que:

Al no implicar vacío el conjunto resultante, luego de analizar la cadena, se está en por lo menos un estado de aceptación. Por lo que la cadena se considera aceptada por el lenguaje del autómata, es el lenguaje regular.

Llegándose a la definición formal:

$$L(A) = \{\omega | \hat{\delta}(x_0, \omega) \ \cap \ X_m \ \neq \emptyset\}.$$

Consideradas las definiciones anteriores, un sistema de eventos discretos es un sistema dinámico de estados discretos, cuya transición de estados se produce a través de ocurrencias, que en general son asíncronas, de eventos, que pueden ser modeladas por un autómata de estado finito. En el caso de procesos industriales, los SED se pueden modelar con AEF determinísticos (Basilio, Carvalho, & Moreira, 2010).

2.3 Diagnosticadores basados en eventos discretos

Una operación de los sistemas industriales actuales requiere contínuos avances en la supervisión y la diagnosticabilidad de las posibles fallas que puedan ocurrir, para asegurar una mejor confiabilidad, seguridad y economía, entre otros. Los sistemas de supervisión y detección de fallas sirven para indicar los valores de las variables de proceso, indicar estados indeseables cercanos a estados no permitidos, para tomar acciones apropiadas con la finalidad de mantener una operación aceptable y evitar daños o accidentes. (Rivera & Manuel, 2007)

Una falla puede definirse como una desviación no permitida, de al menos una propiedad característica de un sistema, o de uno de sus componentes, de su estado o comportamiento normal. (Sayed-Mouchaweh, 2014)

Entonces, un diagnosticador de eventos discretos es un sistema de eventos discretos modelado por un autómata no determinístico, cuya transición de estados es debido a la ocurrencia de estados observables, permite obtener estados estimados del sistema controlado, posiblemente asociados a estados de falla. Para obtener un diagnosticador adecuado, es necesario que el SED del proceso tenga un lenguaje diagnosticable. (Sampath, 1995) expone, en su tesis de doctorado, la definición:

Definición: Un lenguaje vivo y prefijado cerrado se dice que es diagnosticable con respecto a la proyección P y con respecto a la partición Π_f sobre el Σ_f, si se mantiene lo siguiente:

$$\left(\forall i \in \prod f\right)(\exists n_i \in N)\left(\forall s \in \psi(\Sigma_{fi})\right)\left(\forall t \in {}^{L}/_{S}\right) \qquad [\|t\|] \geq n_i \implies D$$

donde, la condición de Diagnosticabilidad D es:

$$\omega \in P_L^{-1}[P(st)] \implies \Sigma_{fi} \in \omega$$

Lo que quiere decir que s puede ser cualquier cadena generada por el sistema que finaliza en un evento de falla del conjunto Σ_{fi}, y, t sea cualquier continuación suficientemente larga de s. Entonces, la condición D requiere que cada cadena pertenezca al lenguaje que produce el mismo record de eventos observables, como la cadena st que debería contener un evento de falla del conjunto Σ_{fi}. Lo que implica que a lo largo de cada continuación t de s se puede detectar la ocurrencia de una falla del tipo F_i con un retardo finito, específicamente en por lo menos en n_i transiciones del sistema después de s.

Dicho de otra manera, la diagnosticabilidad requiere que cada evento de falla lleve a observaciones distintas para permitir la identificación única del tipo de falla con esperas finitas.

Ahora, en el caso de tener múltiples fallas del mismo conjunto, es decir, cuando se tiene más de una falla del mismo tipo, digamos F_i, ocurriendo a lo largo de s del lenguaje L, la definición anterior no requiere que cada una de las ocurrencias sean detectadas. Basta poder concluir que dentro de un número finito de eventos después de la ocurrencia de la primera falla, a lo largo de s, un fallo del conjunto Σ_{fi} ha sucedido.

Esta definición aplica para cualquier lenguaje regular o no-regular, pero que este dentro de $L \subseteq \sum^*$.

Esta definición lleva a establecer los criterios básicos para determinar la existencia de una falla, esto es:

Dentro del alfabeto que maneja un autómata, existen lenguajes asociados a los eventos que puedan ocurrir; de ahí que, un lenguaje específico maneja los eventos de falla, Σ_{fi}, entonces basta con reconocer uno de estos eventos asociado a su etiqueta para determinar que existe falla. Siendo la condición que debe existir un número finito de eventos.

2.3.1 Diagnosticador Centralizado

Este diagnosticador es un autómata observador modificado para operar como diagnosticador, G_d, con un conjunto de eventos conformado por aquellos que son observables del autómata G que modela el proceso, además, cada estado posible dentro del estado global es conformado por una etiqueta que manifiesta si ha ocurrido o no determinado evento σ_f. (Sampath, 1995)

Como indica (Sayed-Mouchaweh, 2014), algunas consideraciones para construir un diagnosticador son:

- El lenguaje generado por el autómata G sea vivo.

- No existe, en el autómata G, ningún ciclo de eventos inobservables.
- Las fallas producen cambios en el funcionamiento del sistema, pero no necesariamente lo llevan a una finalización de operaciones.

Por otro lado, (Sampath, 1995) presenta las características para este tipo de autómata:

- Permite seguir el funcionamiento del sistema, acorde a como está evolucionando en su comportamiento.
- Ante la presencia de alguna falla, esta se puede diagnosticar.
- Los estados posibles, dentro de los estados globales son aumentados con una etiqueta que se asocia al evento de falla suscitado. Esto es:
    - $S_{diag} = (x, l)$; donde $x \in 2^x$ corresponde al estado global del autómata diagnosticador.
    - $l \in 2^{\{\Delta_f\}}$ corresponde a la etiqueta de falla, dentro del plano de fallas posibles.
    - La etiqueta N señala que los eventos de interés no han sucedido en ese estado, pero la etiqueta F_i en cambio señala que se ha dado el evento de interés, y que $e_f \in \sum_f$.
- Considerando que $\sum_f \subseteq \sum_{uo}$, las fallas pueden ser simples o ser un conjunto de eventos de fallas. Pero estas fallas deben cumplir que: $e_f \in \sum_f$.

Entonces, se define al autómata diagnosticador como un autómata determinístico:

$$G_d = (X_d, \sum_d, f_d, x_{0d}, X_{md}),$$

donde:

$x_{0d} = UR(x_0)$,

$X_{md} = \{B \in X_d : B \cap X_m \neq \emptyset\}$,

$f_d(B, e) = UR(\{x \in X : (\exists x_e \in B)[x \in f(x_e, e)]\})$,

Donde el alcance inobservable del estado 'x', $UR(x)$, viene dado por:

$$UR(x) = \{y \in X : \exists t \in \sum_{uo}^{*} [y = f(x, t)]\}$$

Ahora bien, si B resulta ser un conjunto de estados, entonces, el alcance inobservable de B es la unión de los alcances inobservables de cada uno de sus elementos:

$$UR(B) = \bigcup_{x \in B} UR(x)$$

Para construir un diagnosticador centralizado (*observador del SED G para eventos inobservables*), se procede:

i. Definir $x_{0d} = UR(x_0)$. Luego establecer $X_d = \{x_{0d}\}$.

ii. Para cada uno de $B \in X_d$ y $e \in \sum_d$, definir:

$f_d(B, e) = UR(\{x \in X: (\exists x_e \in B)[x \in f(x_e, e)]\})$, siempre que $f_d(x_e, e)$ sea definida por algún $x_e \in B$. Siendo así, agregar el estado $f_d(B, e)$ a X_d. Pero si $f_d(x_e, e)$ no está definida para cualquier $x_e \in B$, entonces $f_d(B, e)$ no está definida.

iii. Repetir el paso ii hasta que toda la parte accesible de G_d haya sido realizada.

iv. Verificar $X_{md} = \{B \in X_d: B \cap X_m \neq \emptyset\}$. (Lafortune & Cassandras, 2008)

Básicamente, suponiendo que existe solo dos etiquetas de falla, N y Y, donde N etiqueta al "estado normal", y Y etiqueta al "estado de falla", la modificación del autómata observador para llevarlo un autómata diagnosticador, G_d, considera lo siguiente (Lafortune & Cassandras, 2008):

1. Cuando se construye el alcance inobservable del estado inicial x_0 de G:
   a) Anexar la etiqueta N a los estados posibles que pueden ser alcanzados desde x_0, para las cadenas inobservables en $[E_{uo} \setminus \{e_d\}]^*$.
   b) Anexar la etiqueta Y a los estados posibles que pueden ser alcanzados desde x_0, para las cadenas inobservables que contienen al menos una ocurrencia de e_d .
   c) Si el estado z puede ser alcanzado ya sea con la ejecución de e_d o no, entonces crear dos entradas en el conjunto de estado inicial de G_d: zN y zY .
2. Cuando se construye una subsecuencia de estados alcanzables de G_d:
   a) Seguir las reglas de la función de transición del autómata observador de G, pero con la forma modificada de arriba, para construir los alcances inobservables con las etiquetas de estado.
   b) Propagar la etiqueta Y. Entonces, cualquier estado alcanzable desde el estado zY tendría la etiqueta Y para indicar que z, y en el proceso de lectura del nuevo estado.
3. En G_d no se definen ningún conjunto de estados marcados.

2.3.2 Diagnosticador Descentralizado

Con lo expresado en el punto anterior, se podría decir que el realizar un diagnosticador centralizado tiene la desventaja principal de conllevar a una complejidad inherente al producto cartesiano del conjunto de estados. Por lo que genera un modelo centralizado que llegaría a ser demasiado grande (en programación) al momento de implementarse.

(Sayed-Mouchaweh, 2014) expone que si se llegare a implementar, este tendría los siguientes inconvenientes:

1. Robustez floja, se da porque un mal funcionamiento parcial del diagnosticador centralizado podría bifurcar en la caída de toda la acción de diagnóstico.
2. Mantenimiento deficiente, debido al hecho de que cualquier cambio en la estructura del sistema podría requerir un rediseño total del diagnosticador, que llegaría a ser muy costoso a más del gran tiempo consumido.
3. Problemas en la comunicación con los sensores, debido que la transmisión de los datos de sensor al diagnosticador centralizado complica el diagnóstico por la necesidad de tener en cuenta el retraso de la señal y los datos faltantes, como también los errores durante su transmisión al punto central.

Para esto se tiene el criterio de la "Codiagnosticabilidad", siendo la idea principal:

"Cada uno de los sensores de un sistema será monitoreado por un diagnosticador propio, el que no tendrá comunicación con ningún otro del sistema completo. Pero que a través de un Coordinador podrá compartir información de manera limitada. Pero será el Coordinador quien en función de unas directivas específicas tome la decisión, en función de la información que le llegue de todos los diagnosticadores locales."

Además de exponer la idea de lo que es Codiagnosticabilidad, (Sayed-Mouchaweh, 2014) manifiesta que las posibles fallas de los sensores se pueden determinar a partir de diagnosticadores locales. Los cuales no tendrían comunicación directa entre ellos, sino que se encargarían de chequear a sus sensores respectivos y transmitir al coordinador las novedades encontradas. Pero para impedir algún problema de ambigüedad, se permite una comunicación limitada, por intermedio del coordinador, entre los diagnosticadores locales.

Entonces, un sistema codiagnosticable se puede convertir en uno diagnosticable, pero un sistema diagnosticable no se puede asegurar que sea codiagnosticable.

Para poder entender lo que significa lo planteado hasta ahora, se da los siguientes conceptos:

Diagnosticabilidad local: en un componente o subsistema i, al ocurrir una falla f, si sólo se puede diagnosticar por un diagnosticador local G_{di} se dice que es diagnosticable localmente, sin la necesidad de comunicarse con otros diagnosticadores locales.

Diagnosticabilidad independiente: en un componente o subsistema i, al ocurrir una falla f, si esta falla es diagnosticable en forma independiente por al menos uno de los diagnosticadores locales, además, no se requiere que exista comunicación entre ellos.

La principal diferencia entre diagnosticabilidad local y diagnosticabilidad independiente es que en la anterior, la ocurrencia de la falla en el componente i será diagnosticada por el diagnosticador G_{di} asociado a ese componente, mientras que en la última la falla puede ser diagnosticada por cualquier otro diagnosticador G_{dj} con $j \neq i$. Por lo que la noción de diagnosticabilidad requiere de la disponibilidad de un modelo local.

Diagnosticabilidad descentralizada: la ocurrencia de una falla f es diagnosticada con certeza y, con un retardo de tiempo finito por la cooperación entre los diagnosticadores locales. Entonces, se dice que la falla es diagnosticable descentralizada.

Esto nos lleva a plantear dos formas de manejar el diagnóstico descentralizado, que se lo puede encontrar en detalle en (Sayed-Mouchaweh, 2014). La primera es el manejo de un modelo global para determinar el tipo de diagnosticador a desarrollar, en donde el sistema requiere de un coordinador simple basado en un conjunto de reglas para llegar a ser codiagnosticable. (Ver Figura 2. 7)

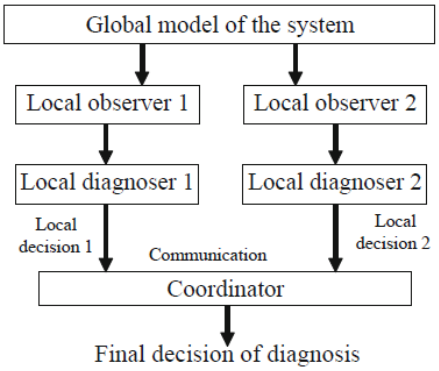


Figura 2. 7 *Estructura de un diagnosticador descentralizado*
Fuente: (Sayed-Mouchaweh, 2014)

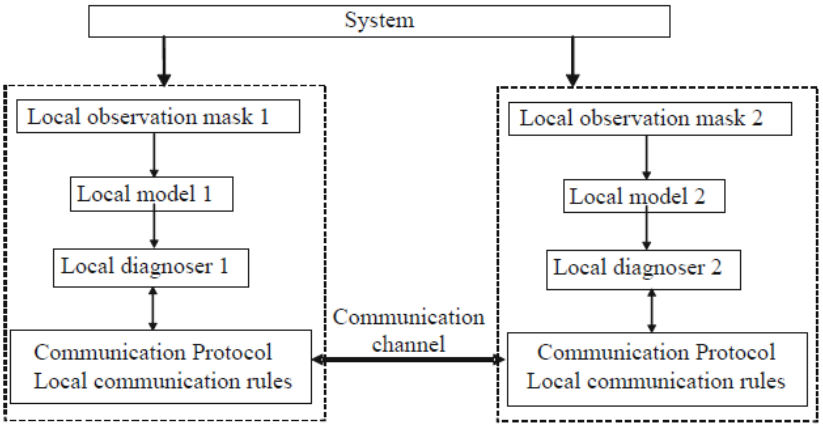


Figura 2. 8 *Estructura de un diagnosticador distribuido, para dos sitios*
Fuente: (Sayed-Mouchaweh, 2014)

Entre tanto, la segunda no requiere de un modelo global, ya que trabaja en base a ciertas especificaciones, que conjuntamente con la composición paralela de los modelos planteados, se logra llegar al diagnosticador adecuado. (Ver Figura 2. 8)

Algunos ejemplos de cómo desarrollar un diagnosticador descentralizado se puede encontrar en (Basilio, Carvalho, & Moreira, 2010) y (Lafortune & Cassandras, 2008).

2.3.2.1 Diagnosticador modular en sistemas de eventos discretos

Con lo anterior planteado se tiene dos ideas:

1. El tener un solo diagnosticador que tomaría la tarea de verificar a todo el sistema de forma centralizada.
2. Por otro lado, si se fracciona en subsistemas de diagnóstico agrupando algunos de los dispositivos a verificar, se estaría bajando el coste computacional de implementar un diagnosticador centralizado, y se daría mayor flexibilidad a dicho sistema.

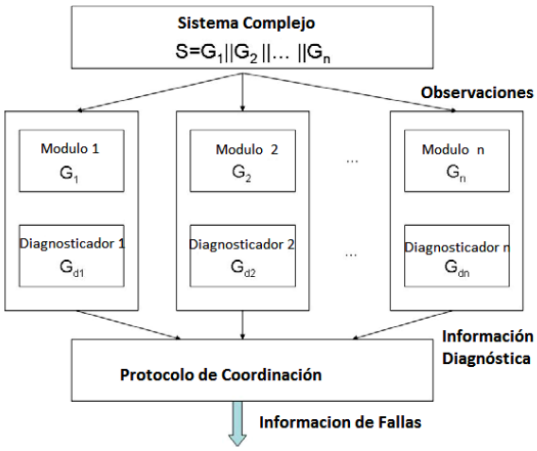


Figura 2. 9 *Estructura de un diagnosticador modular.*
Fuente: (Monsalve, 2011)

Los enfoques modulares, Figura 2. 9, consideran que un sistema complejo modelado por un autómata G, se puede fraccionar en módulos que se componen por medio de autómatas G_i (Monsalve, 2011):

$$G = G_1 \parallel G_2 \parallel \cdots \parallel G_n,$$

Al momento de diseñar los diagnosticadores para cada módulo M_i, se debe cuidar que cada módulo contenga un dispositivo común a ellos. En la Figura 2. 10 se muestra un sistema en cual se tiene un actuador y dos sensores. Entonces, cada modulo M_i estaría formado por un autómata dado por $M_i = G_{ci} \parallel S_i$; de lo cual se tiene que:

- G_{ci} es el controlador del sistema.
- $S_i = \overset{\parallel}{}_j G_j$ es la composición paralela de los autómatas de los componentes del módulo.

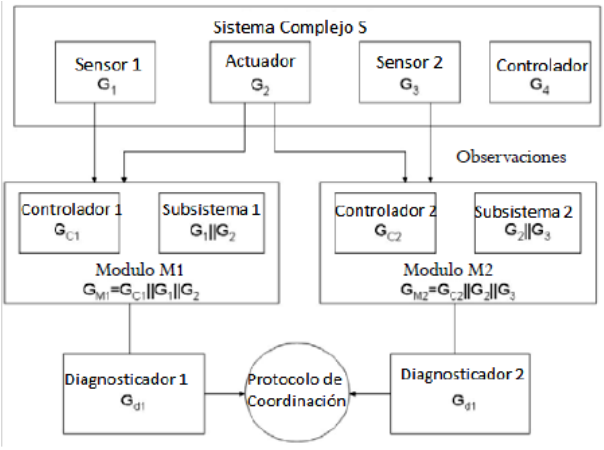


Figura 2. 10 *Organización de un diagnosticador modular*
Fuente: (Cerrada, Ferrarini, & Dedé, 2010)

Teniéndose que cada $M_i = G_{ci} \parallel S_i$, se genera un autómata $M_i = (X_i , \sum_i , f_i , x_{0i})$, donde X_i es el conjunto de estados, $\sum_i$ es el conjunto de eventos del módulo, f_i es la función de transición, y x_{0i} es el estado de inicio. (Monsalve, 2011)

(Cerrada, Ferrarini, & Dedé, 2010) señalan que una arquitectura así, presenta las características:

i. El sistema se lo considera un conjunto de componentes físicos, relacionados por otro que interactuarían juntos. Este elemento común es el actuador.

ii. Cada subsistema es compuesto por dos o más elementos, incluyendo el común.

iii. Un componente de sincronización, derivado del controlador monolítico, es considerado para cada subsistema. Este incluirá la transición asociada con la operación nominal y lo relacionado a las situaciones no seguras.

Entonces, para cada uno de los subsistemas (módulos) se define una partición de fallas Δ_f^n (con $n = 1, 2, ...$) como un conjunto que contiene todas las etiquetas de falla de cada elemento; estos no son disjuntos entre sí.

Ahora bien, para considerar un tipo de *algoritmo de diagnóstico* con un análisis pasivo temporizado, el mismo que es propuesto en (Ferrarini, Brusa, & Veber, 2008), se debe tener presente que:

Definición de tiempo de espera.- Siendo $c \in \sum_c$ un evento controlable; el correcto funcionamiento de un sistema se considera si luego de haber ocurrido un evento c, dentro de un tiempo T_{max} se verifican todos los demás eventos esperados. Esto conlleva a pensar en que el sistema está en un estado libre de falla, estado nominal.

Definición de estado de equilibrio.- Siendo $o \in \sum_o$ un evento observable; entonces, al no verificarse más eventos observables después de un T_{max} , se asume que el sistema está en un estado de equilibrio.

Para poder realizar un análisis temporizado, existen algunos estados transitorios del autómata de cada módulo M_i, que están identificados con la extensión $-t$. Esto es, $X_i = X \cup X_t$ donde X_t son los estados transitorios.

Conjuntamente (Cerrada, Ferrarini, & Dedé, 2010) y (Monsalve, 2011), plantean:

- En el estado de equilibrio, los estados estimados del diagnosticador sólo se restringen a los estados que están sin la etiqueta -*t*, es decir, el estado de equilibrio es un subconjunto de cada estado del diagnosticador.

- Nuevos eventos observables desde el actual estado de equilibrio están relacionados con una nueva dinámica transitoria que permita llegar a un nuevo estado de equilibrio. Existe una relación fuerte entre el estado de equilibrio actual y los anteriores que proporciona más información en el análisis diagnóstico.

Siendo G_d el diagnosticador para el sistema S, descrito por $G_d = (X_d, \sum_o, f_o, x_o)$, con las definiciones comunes de la tupla, y, con x_d como un estado de G_d definido como el conjunto $x_d = \{(x_1, l_1), \dots, (x_m, l_m)\}$, donde $x_i \in X$ y $l_i \in \Delta$. En donde $\Delta = 2^{\{\Delta_f\}} \cup \{N\}$ que es el conjunto de todas las fallas posibles; y, Δ_f es el conjunto de todas las posibles fallas.

Entonces:

- ✓ En G_d, el estado de equilibrio está definido por el conjunto dado por $x_{eq} \subset x_d \in X_d$. Tal que $x_{eq} = \{(x_i, l_i) | x_i \notin X_t\}$.
- ✓ Siendo $x_{eq-j} \subset x_j$ como el estado actual de equilibrio del diagnosticador G_d en el tiempo de espera, y, $o \in \sum_o$ como un evento observable. El estado de equilibrio sucesivo x_{eq-j^+} de G_d es definido como el conjunto compuesto por el conjunto alcanzable debido al evento o, que está dado por

$$S(x, \sigma_0) = \{\delta(x, \sigma_0), \sigma \in (\sum_{uo} \cap \sum_{uc})\}$$

Advertir que $x_{eq-j^+} \subset x_{j^+} \in X_d$, sabiendo que x_{j^+} es el estado sucesivo de G_d que ha sido alcanzado desde x_j por la acción del evento o.

- ✓ Luego, por el tipo de análisis temporizado, el estado de equilibrio $x_{eq-j^+} \subset x_{j^+} \in X_d$ se dice F_i cierto si $\forall (x, l) \in x_{eq-j^+}, F_i \in l$.

(Cerrada, Ferrarini, & Dedé, 2010) y (Monsalve, 2011) establecen el protocolo de comunicación y las reglas de coordinación entre diagnosticadores locales. Siendo:

Protocolo de comunicación:

Al tener n diagnosticadores locales, con eq_j (donde $j = 1, 2, \dots, n$) como la representación de los estados de equilibrio posibles presentes en cada uno de los diagnosticadores. En donde, el estado global de equilibrio que proporciona la información equivalente a un diagnosticador monolítico G_d, está formado por el conjunto $X_{GS} = \{(\hat{x}_1, \hat{l}_1), (\hat{x}_2, \hat{l}_2), \dots, (\hat{x}_n, \hat{l}_n)\}$.

Este conjunto se obtiene por la función *merge (mezcla)*, definida como:

$$merge[(x_1, l_1), \dots, (x_n, l_n)] = \begin{cases} (\hat{x}, \hat{l}) : \hat{x} \in x_n ; \hat{l} \in l_n \quad si \ \exists x_c \in x_i \ \forall x_i \\ no\ definido\ en\ otro\ modo \end{cases}$$

Para poder efectuar esta función, los estados posibles de cada diagnosticador deben compartir un estado x_c correspondiente al elemento común.

Una vez obtenida la etiqueta de falla, como F_i cierta, se debe propagar a todos los diagnosticadores de cada subsistema, para encontrar otras posibles fallas. Para esto, se emplea la función:

$$match[eq_j, F_k] = \begin{cases} (\tilde{x}, \tilde{l}) \in eq_j : \ F_k \in \tilde{l} \ \ \forall (\tilde{x}, \tilde{l}) \in eq_j \\ no\ definido\ en\ otro\ modo \end{cases}$$

En conclusión: *cualquier falla se considera F cierta, cuando F pertenece a* $\tilde{l}$*, para todo* $(\tilde{x}, \tilde{l})$ *que pertenece a* $\tilde{X}_S$.

Reglas de coordinación:

i. Se asume el estado actual de equilibrio $x^i_{eq-j^+}$ del diagnosticador $G^i{}_d$ es F_k cierta, y su correspondiente etiqueta F_k se transmite al coordinador. Luego, se aplica la función $match$ con los estados de equilibrio de los otros diagnosticadores, en donde, una falla F_i se considera aislada si en $\tilde{X}_S$ resulta F_i cierta. Esto para $\forall (\hat{x}, \hat{l}) \in \tilde{X}_{GS}$, $F_i \in \hat{l}$.

ii. Siendo $x^i_{eq-j^+}$ el último estado de equilibrio observado después del tiempo de espera, y, x^j_{eq} el estado de equilibrio de los otros diagnosticadores, antes de que se consuma el tiempo de espera. Entonces, una falla $F_i \in \prod_f{}^j$ se puede aislar,

si posteriormente a la aplicación de $merge$ entre $x^i_{eq-j^+}$ y x^j_{eq} , se tiene que $\forall(\hat{x},\hat{l}) \in X_{GS}$, $F_i \in \hat{l}$.

iii. Si falla alguna no es aislada, tomar el x^i_{eq-j} con el x^j_{eq} y aplicar la función $merge$. Una falla F_i contenida en la partición de fallas $\Pi^j{}_f$ podría ser aislada si después de aplicar *merge* entre x^i_{eq-j} y x^j_{eq} , resulta $\forall(\hat{x},\hat{l}) \in \hat{X}_{GS}$, $F_i \in \hat{l}$. Considerando que

$$\hat{X}_{GS} = \{(\hat{x},\hat{l}) \in X_{GS} | sea\ la\ última\ observación\ definida\}.$$

<u>*Algoritmo de coordinación*</u>, derivado de las reglas de coordinación presentadas:

- Por la dinámica del sistema en cuestión, se plantea un $T_{máx}$ adecuado.
- Dado un evento observable del sistema, se ejecutan los diagnosticadores locales, y detectar un *timeout*. Al producirse un *timeout* ejecutar:

1. Determinar los subconjuntos de equilibrio actuales $x^i{}_{eq-j} \subset x^i{}_j$ en cada diagnosticador.
2. Para cada estado estimado $x^i{}_{eq-j}$, determinar si F_K es F-cierta, para todo F_K que pertenezca a Δ_f.
   2.1. Si alguna falla es $F_K - cierta$ aplicar la función *match* con el resto de los estados de equilibrio, para aislar nuevas fallas, conforme la regla de coordinación (i).
   2.2. Si ninguna falla es aislada aplicar *merge*, y aislar nuevas fallas conforme a la regla de coordinación (ii) o (iii).
   2.3. Si alguna falla es detectada, transmitir la información al coordinador, e ir al paso 2.1.
3. Si ninguna falla es detectada, esperar por un nuevo estado observable , y, para el correspondiente diagnosticador:
   3.1. Determinar el estado de equilibrio sucesivo $x^i{}_{eq-j^+} \subset x^i{}_{j^+}$, considerando de $x^i{}_{j^+}$ los estados estimados que permiten evolucionar a la siguiente transición.
   Para los correspondientes estados estimados de $x^i{}_{eq-j^+}$ volver a los pasos 2 y 3, sucesivamente.

En esta última parte se establece una diferencia con (Monsalve, 2011), esto es que en vez de quedarse esperando el evento que permita avanzar en la búsqueda de la falla, que podría tomar mucho tiempo; se excita al sistema en cuestión con un evento, y se repite dicho análisis, para dar con la falla. Siendo este el procedimiento que se aborda en los ejemplos que se muestran más adelante.

CAPÍTULO 3

SISTEMA DE TRANSPORTE Y DISTRIBUCIÓN DE PRODUCTOS EN BANDAS TRANSPORTADORAS

3.1 Conceptualización

Con el avance de la producción, se hizo importante contar con un medio de transporte de material que asegure la continuidad de ciertos procesos industriales. De ahí que en 1795 fue cuando la cinta transportadora se convirtió en un medio popular para el transporte de materiales a granel. (Salinero, 2013)

Figura 3. 1 *Cinta de transporte de finales del siglo XIX. Deutsches - Museum*
Fuente: (Salinero, 2013)

Las primeras bandas fueron muy simples (ver Figura 3. 1), constituidas por una cama plana en la cual se movía una cinta de cuero o lona, y sobre ella era colocado material sólido, por ejemplo piedras, tierra, carbón, etc., llevado a grandes distancias, Pero tenían la desventaja del desgaste muy rápido. Con el tiempo se usó un revestimiento de goma sobre el material de la banda, esto le daba más durabilidad. Este invento era el de Thomas Robins.

En la actualidad se emplean en las más variadas aplicaciones, logrando establecer líneas de producción en casi cualquier tipo de industrias. Y con el acople del motor eléctrico se ha logrado darles mayor capacidad de carga.

3.2 La cinta de transporte

Dado que estas cintas transportadoras se usan para el transporte de material a granel, ya sea en posición horizontal o inclinada, presenta las ventajas:

i. Una velocidad alta de transporte.

ii. Puede llevar material a grandes distancias.

iii. El transporte de material es de manera continua. (Salinero, 2013)

Por otro lado, la banda (independiente del material usado) es la encargada de:

- Contener al material a transportar.
- Transmite la fuerza requerida para el transporte.

Los ramales superior y de retorno descansan sobre un conjunto de rodillos colocados en una armazón metálica (ver Figura 3. 2). En los dos extremos del transportador, la banda se enrolla en tambores, unos de los cuales, acoplado a un órgano motor, transmite el movimiento. (Salinero, 2013)

Más información se puede encontrar en (Maquinariaspesadas.org, 2016) y (Salinero, 2013)**.**

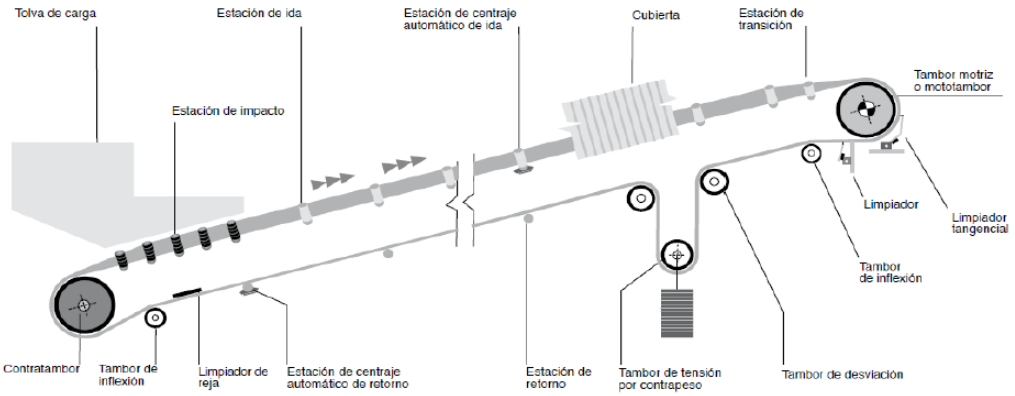


Figura 3. 2 *Diagrama de la cadena de una cinta de transporte*
Fuente: (Salinero, 2013)

3.3 Descripción del Proceso de aplicación (actuador y sensores)

Para explicar este caso de análisis, se tomará un ejemplo de uso de bandas transportadoras empleado en la industria, el transporte de producto de un lugar a otro, con la finalidad de cumplir ciertos procesos de selección. Estos procesos se definen de acuerdo al mecanizado, calidad, estándares de producción, entre otros.

Cuando se trabaja en una línea de producción, pequeñas fallas o indecisiones conllevan a atascos del sistema de transporte o envío del producto por bandas no correctas, lo que puede generar pérdida de tiempo en el proceso de producción. Es por ello que el análisis de fallas de este tipo de componentes en un sistema de manufactura es importante.

Para empezar (ver Figura 3. 3), se puede colocar el producto en la banda 1 o en la banda 2. Si se coloca en la banda 1, la banda 2 no puede funcionar, y viceversa. Para el accionamiento se tiene un pedal para cada una, de inicio del transporte. Los sensores S1 y S2, sensores fotoeléctricos, son los encargados de enviar la señal de detección de producto al PLC para activar la banda 3, entre tanto, la detención de la banda 1 o banda 2 se hace por tiempo (timer de software en el PLC), luego de depositar la carga en la banda 3.

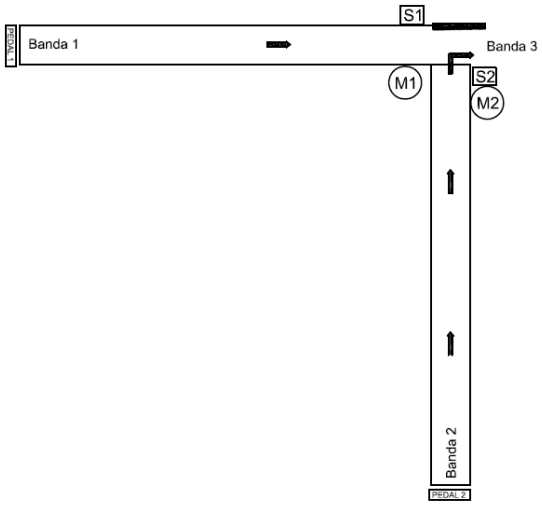


Figura 3. 3 *Partes principales de una banda transportadora.- proyecto.*
Fuente: Autor

Para efectuar la selección se tiene el siguiente arreglo de bandas:

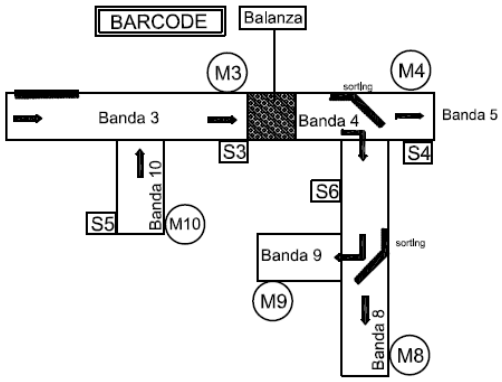


Figura 3. 4 *Proceso de selección de material de acuerdo a estándares de producción.*
Fuente: Autor

En esta parte y acorde a la Figura 3. 4, el lector de código de barras (*barcode*, por su indicación en inglés) es el encargado de identificar el tipo de producto según una etiqueta de marcación. Entonces, la operación de los sensores y bandas son:

S3: sensor fotoeléctrico, detecta el producto que viene desde la banda 1 o banda 2, y envía la señal al PLC para que se accione un temporizador para la detención de la banda 3, no sin antes colocar el producto para el pesado en la balanza y activar a la banda 4.

S4: sensor fotoeléctrico, detecta al producto y en envía la señal al PLC, indicando salida del producto de la banda 4, y activa la banda 5, para continuar el producto a la salida de este proceso.

S5: sensor fotoeléctrico, detecta la presencia de producto en la banda 10, y envía la señal al PLC para que se active, y la apague luego de un tiempo, por la acción de un temporizador en el programa. Esto en el caso que el operario envíe producto por esta banda para reprocesarse. Pero se debe considerar el bloqueo de las bandas 1 y 2, al darse esta alimentación de producto.

S6: sensor fotoeléctrico, detecta el producto y envía la señal al PLC, quien detiene la banda 4 después de un tiempo, y, activa la banda 8. Tanto la banda 8 y 9 se detienen por la acción de un temporizador.

Se tiene dos compuertas que son las encargadas de conducir el producto. Para cada compuerta, la activación es electroneumática, contando con un cilindro de doble efecto controlado por una electroválvula 4v/2p. En el cilindro se cuenta con dos sensores inductivos de posición (anterior y posterior), para determinar la posición del vástago. Su control se lo realiza desde el software del PLC (ver Figura 3. 5).

La balanza es la encargada de pesar el producto y envía esa lectura al PLC, en cual, con la lectura del tipo de producto dado por el lector de código de barras, se busca en la base de datos los rangos de tolerancia. Si el producto está en rango y es de "reposición", se le permite seguir a la banda 5, banda 6 y banda 7. (Ver Figura 3. 6)

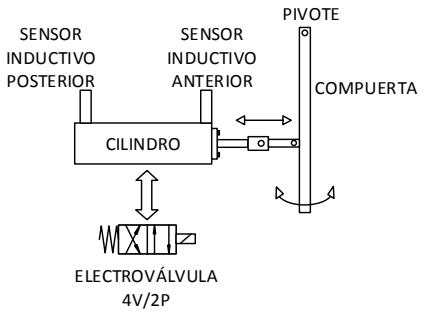


Figura 3. 5 *Control electroneumático de las dos compuertas para clasificar material*.
Fuente: Autor

A la salida de la banda 3, si el peso está en rango y es "producto original", se activa la primera compuerta, desviando el producto hacia la banda 8, para el trámite correspondiente; pero, si existe alguna observación, se envía por la banda 9, activando la segunda compuerta, para que el operario pueda determinar lo sucedido, y si es el caso, envíe nuevamente el producto por la banda 10 a que se reprocese. El sensor 5 (sensor fotoeléctrico) indica la presencia del producto y envía al PLC la señal correspondiente para que se active y llevarlo hasta la banda 3, siempre y cuando las bandas 1 y 2 no estén transportando producto. El apagado de la banda 9 es por un timer del programa de control en el PLC. La activación de la banda 10 bloquea a las bandas 1 y 2. (Ver Figura 3. 4)

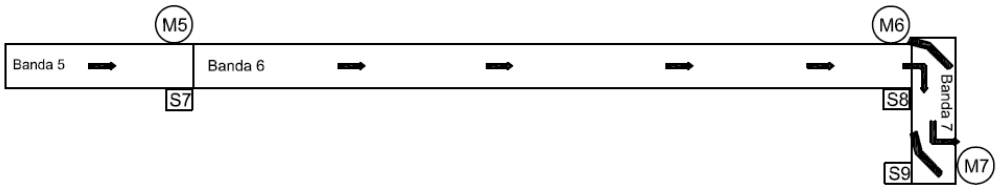


Figura 3. 6 *Bandas de transporte de producto.*
Fuente: Autor

Para concluir el traslado (ver Figura 3. 6) del producto en condiciones de continuar, se tiene las bandas 5 – 6 – 7; los sensores S7 – S8 – S9 (sensores fotoeléctricos) tienen la finalidad de detectar la presencia del producto, y enviar la señal correspondiente al PLC para impulsar la detención temporizada de la banda anterior y activación de la banda siguiente.

Cada una de las bandas tiene acoplado un motor trifásico, que para su comando se usa guardamotores. El PLC, en el que está corriendo el software, debe realizar el control de todo el proceso.

Entonces, la tendencia es a elaborar un programa de control (según el equipo que se tiene) para interactuar con los sensores y actuadores presentes. Pero, por lo general, no se elabora un algoritmo que verifique las posibles fallas de estas unidades. A saber:

- Los motores pueden detener su funcionamiento por sobrecarga.
- Los sensores, eventualmente por maltrato de los operarios (*rotura de cable, desconexión, daño físico, etc.*), podrían no dar señal; causando un atasco en el proceso de producción.
- Las electroválvulas que manejan las compuertas, pueden quedarse atascadas en la apertura o cierre.
- Los cilindros podrían atascarse por fallas mecánicas y no completar su recorrido operativo.

3.3.1 Proceso de análisis (actuador y sensores)

Al ser un proceso largo de producción, se escoge la parte de alimentación para efectuar este análisis, descrito en la Figura 3. 3, con las condiciones de operación siguientes:

- El producto puede ingresar al proceso ya sea por la banda 1 o la banda 2.
- Ambas bandas son mutuamente excluyentes en su tiempo de activación.
- Cuando el producto llega al final de cualquiera de ellas, S1 o S2 disparan el timer de apagado en el software de control general, y a la vez ponen en marcha a la banda 3.

Como ya se mencionó, cada banda tiene un motor del tipo trifásico que es activado por un contactor de potencia que sería el "actuador".

Para monitorear cuándo entra en funcionamiento el motor dispone de tres sensores:

- Dos sensores acoplados al actuador, que son complementarios y solidarios, para conocer cuando se activa o desactiva el motor. Estos suelen ser del tipo de contactos acoplados al contactor.

Ahora bien, el contactor de potencia (Ver Figura 3. 7), es un dispositivo denominado también "telerruptor", dado que permite interrumpir o permitir el paso de corriente eléctrica a distancia. Su funcionamiento se basa en un circuito electromagnético conformado por una armadura y una bobina eléctrica. (Ver Figura 3. 8)

Figura 3. 7 *Presentación comercial de un telerruptor SIEMENS 3RT2016-1BB42 SIEMENS*
Fuente: http://static2.tme.eu/products_pics/c/b/f/cbf7300f5ab16237dd094e92439d7241/386914.jpg

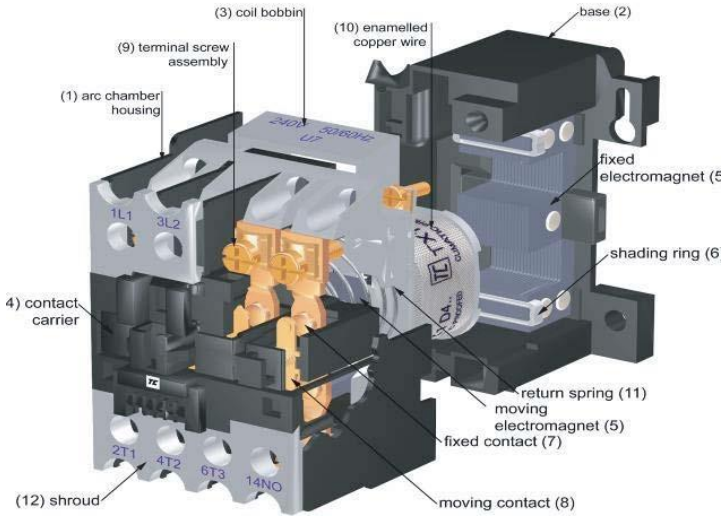


Figura 3. 8 *Partes constitutivas de un telerruptor, denominado también contactor.*
Fuente: http://www.thermalinc.com/power/pics/Contactor_EEC.jpg

A este contactor se le acopla un bloque de contactos auxiliares (Figura 3. 9) para que sean los sensores, un contacto cerrado y un contacto abierto.

Figura 3. 9 *Montaje de contacto auxiliares frontales: 2 NO + 2NC.*
Fuente: http://www.dasautomatica.com/catalog_icon/955b3581187bca0584e0da4cc4910931.jpg

Estando en un circuito eléctrico, Figura 3. 10, permite controlar el encendido o apagado de una carga inductiva (motor eléctrico), en este caso cada banda transportadora posee uno de estos.

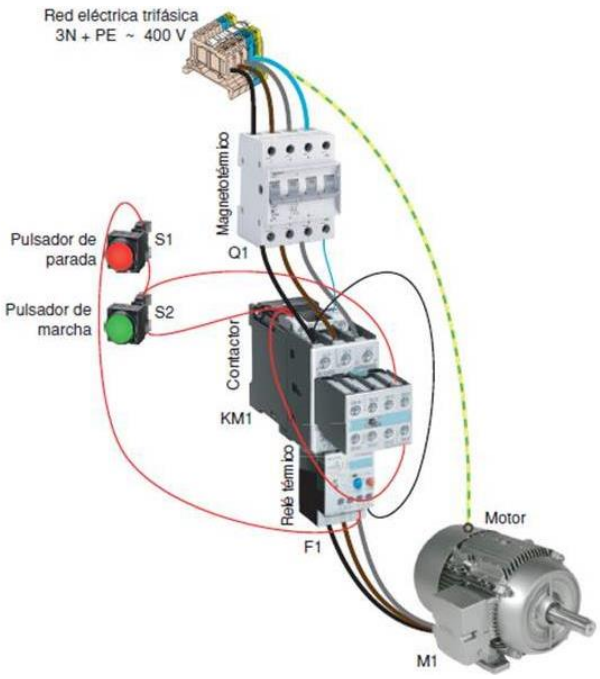


Figura 3. 10 *Conexión eléctrica de un contactor para el control de un motor trifásico.*
Fuente: https://automatismoindustrial.files.wordpress.com/2012/10/esq1.jpg

Otro sensor considerado es el sensor de rotación colocado en la carga, cerca del eje, para saber si está o no en rotación (ver Figura 3. 11). En el mercado existen una gran variedad de sensores y equipos que permiten chequear que el motor este en movimiento. Tal es el caso, del equipo "Electronic Rotation Control" de la Empresa Fargo Control INC.

(Fargo Control, 2017) en la hoja de datos para este equipo indica que este sensor mide el intervalo de tiempo entre dos señales de un sensor, en un mecanismo giratorio o de ciclo. Cuando el intervalo de tiempo se vuelve mayor que el valor preajustado, el dispositivo señala una velocidad o parada completa del movimiento del mecanismo.

Los controles de rotación vienen en dos tipos CRTP, que tiene los controles separados y requiere un sensor o CRT30 que tenga los sensores y los controles en una carcasa.

Estos equipos son ideales para el control de correas de transporte deslizantes, rotura de cadena, interrupciones de energía o sobrecargas, etc. Las aplicaciones incluyen transporte de correa, elevadores de cadena, molinos, amoladoras, bombas, mezcladores, y otras situaciones en las que el control de movimiento de cíclico es necesario.

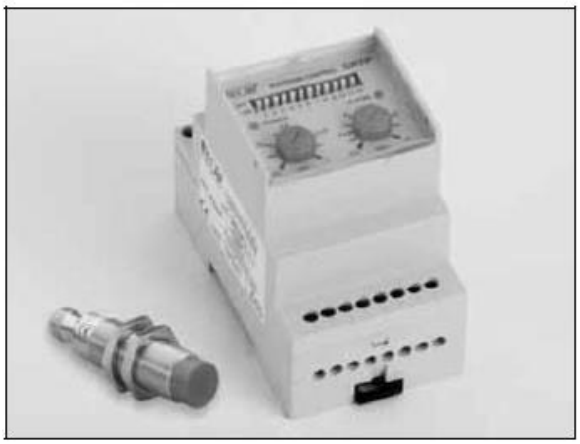


Figura 3. 11 *Equipo de medición de velocidad, CRTP de FARGO, en la foto: unidad de control electrónico y sensor externo CRT30*
Fuente: http://www.fargocontrols.com/pdf/controls/Fargo-RC_N.pdf

Este dispositivo permite establecer una conmutación de un contacto para indicar si el eje del motor está girando o no. Esta señal se introduce al PLC del sistema y se establece la dinámica del control. Dado esto, este contacto es el sensor de movimiento del motor.

La operación se basa en la colocación, en el eje del motor, una rueda de ribbon con 4 placas de acero en forma de cruz. Estas permitirán captar pulsos por el sensor externo, basados en el principio de operación de los sensores inductivos, que dice que:

Los sensores inductivos son usados para realizar la detección, sin contacto previo, de parte metálicas. Entonces, la presencia de un objeto metálico provoca una amortigüación de la amplitud de oscilación, estas elevaciones o caídas de oscilación se identifican por un circuito de umbral que cambia la salida del sensor. (Fargo Control, 2017)

Para mayor información se puede consultar la página web del fabricante.

Ahora, el motor trifásico (ver Figura 3. 12) es una carga inductiva que aprovecha el campo electromagnético creado en sus bobinas para hacer girar un rotor o flecha. Esta característica es dada por el tipo de corriente alterna que es alimentada a dichas bobinas.

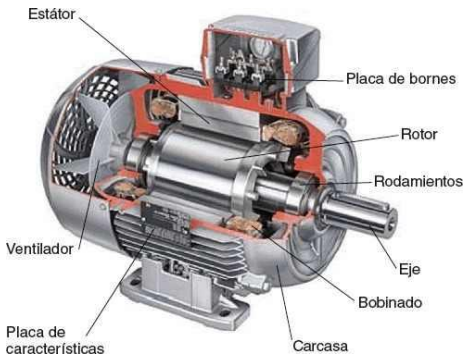


Figura 3. 12 *Partes de un motor trifásico de jaula de ardilla.*
Fuente: https://http2.mlstatic.com/motor-trifasico-75hp-1500rpm-55-kw-siemens-chierro-D_NQ_NP_453601-MLA20363652212_072015-O.jpg .

Estos motores se ensamblan directamente con cajas reductoras, que sirven para disminuir la velocidad de rotación a valores que permitan a las bandas transportadoras, llevar la carga de manera segura. Entonces, una caja reductora (ver Figura 3. 13) es un conjunto de engranajes, lubricados por aceite, que bajan la velocidad de rotación de entrada, a una velocidad de rotación de salida que depende de un factor de reducción dado por el fabricante de la caja reductora.

Figura 3. 13 *Partes de una caja reductora.*
Fuente: http://www.marioperez.com.mx/wp-content/uploads/2015/05/reductor-cicloidal.jpeg

Cabe indicar que para evitar desgaste mecánico de las partes, el mecanismo interno se mueve en una laguna de aceite, que lubrica y refrigera los engranajes.

3.4 Modelo de los actuadores y sensores

El actuador y los sensores del motor trifásico para el movimiento de las bandas transportadoras se consideran como componente de dinámica binaria, cuya dinámica de funcionamiento conjunta representan sistemas de eventos discretos que pueden ser modeladas por AEF. Las ocurrencias de fallas en dichos componentes se consideran como eventos no observables que se incorporan al modelo de AEF, a partir del cual se pueden construir diagnosticadores de fallas de eventos discretos.

Los modelos se construyen siguiendo las indicaciones dadas en (Sampath, 1995) y (Lafortune & Cassandras, 2008), para lo cual se debe considerar las descripciones dadas en la Tabla A. 1, referente a la nomenclatura utilizada; entre tanto para la codificación de las fallas referirse al Tabla A. 2. Además, la descripción de cada modelo de autómata se expresa de la siguiente forma:

$$x_0 = \{estado\ inicial\}$$
$$X = \{conjunto\ de\ estados\ del\ AEF\}$$
$$\Sigma_o = \{conjunto\ de\ eventos\ observables\}$$
$$\Sigma_{uo} = \{conjunto\ de\ eventos\ inobsevables\}$$
$$\Sigma_c = \{conjunto\ de\ eventos\ controlables\}$$
$$\Sigma_{uc} = \{conjunto\ de\ eventos\ no\ controlables\}$$
$$\Sigma_f = \{conjunto\ de\ eventos\ de\ falla\}$$

De acuerdo a lo descrito en la sección anterior, el actuador de un motor es el contactor de potencia denotado como K, y los sensores serían dos de tipo contacto denotados como SK, y un sensor de movimiento denotado como SM.

Definido esto, los modelos de AEF para los actuadores K_1, K_2 y K_3 para cada motor, son:

Modelado del actuador 1: K1

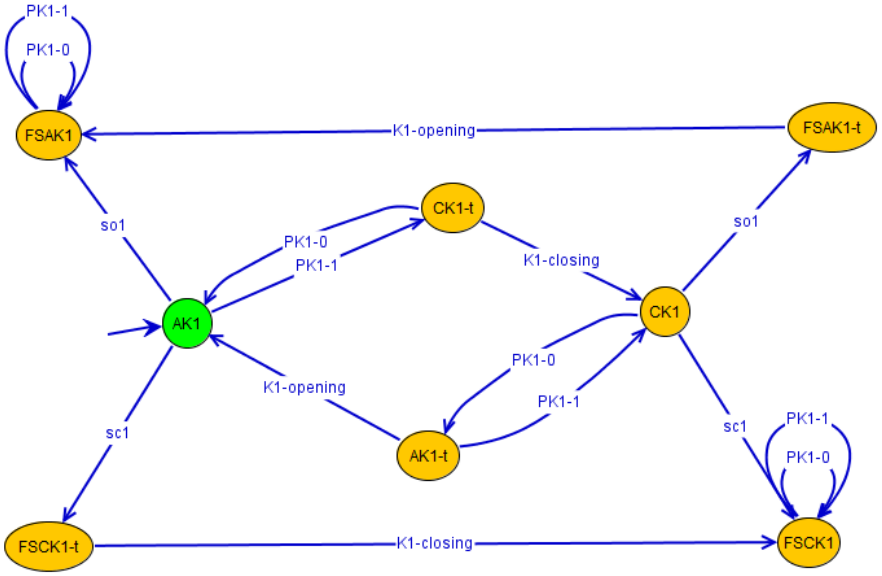


Figura 3. 14 *Modelo de FSM para el actuador 1: K1. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{AK1\}$

$X = \{AK1, CK1 - t, CK1, AK1 - t, FSAK1, FSAK1 - t, FSCK1, FSK1 - t\}$

$\Sigma_o = \{PK1 - 0, PK1 - 1\}$

$\Sigma_{uo} = \{so1, sc1, K1 - closing, K1 - opening\}$

$\Sigma_c = \{PK1 - 0, PK1 - 1\}$

$\Sigma_{uc} = \{so1, sc1, K1 - closing, K1 - opening\}$

$\Sigma_f = \{so1, sc1\}$

Modelado del actuador 2: K2

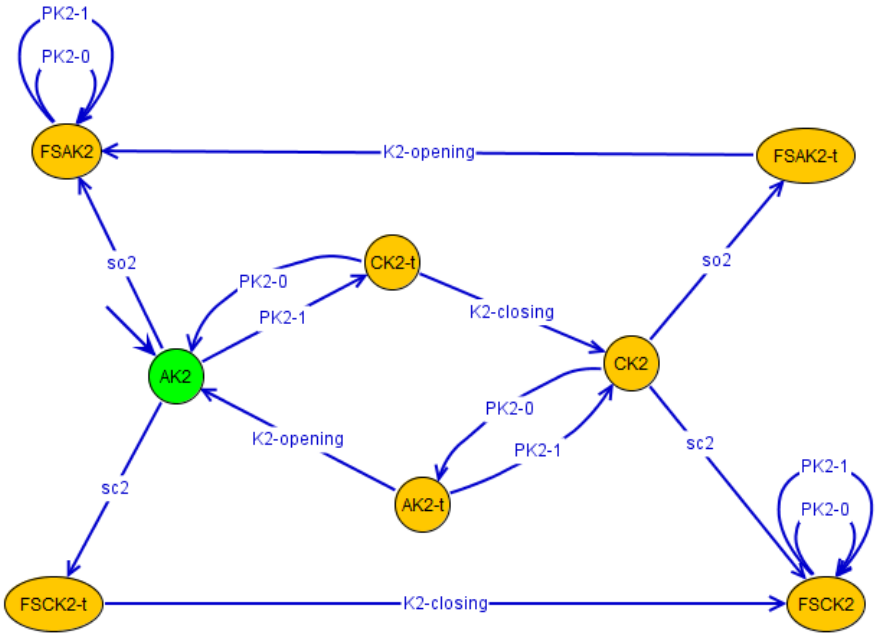


Figura 3. 15 *Modelo de FSM para el actuador 2: K2. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{AK2\}$

$X = \{AK2, CK2 - t, CK2, AK2 - t, FSAK2, FSAK2 - t, FSCK2, FSK2 - t\}$

$\Sigma_o = \{PK2 - 0, PK2 - 1\}$

$\Sigma_{uo} = \{so2, sc2, K2 - closing, K2 - opening\}$

$\Sigma_c = \{PK2 - 0, PK2 - 1\}$

$\Sigma_{uc} = \{so2, sc2, K2 - closing, K2 - opening\}$

$\Sigma_f = \{so2, sc2\}$

Modelado del actuador 3: K3

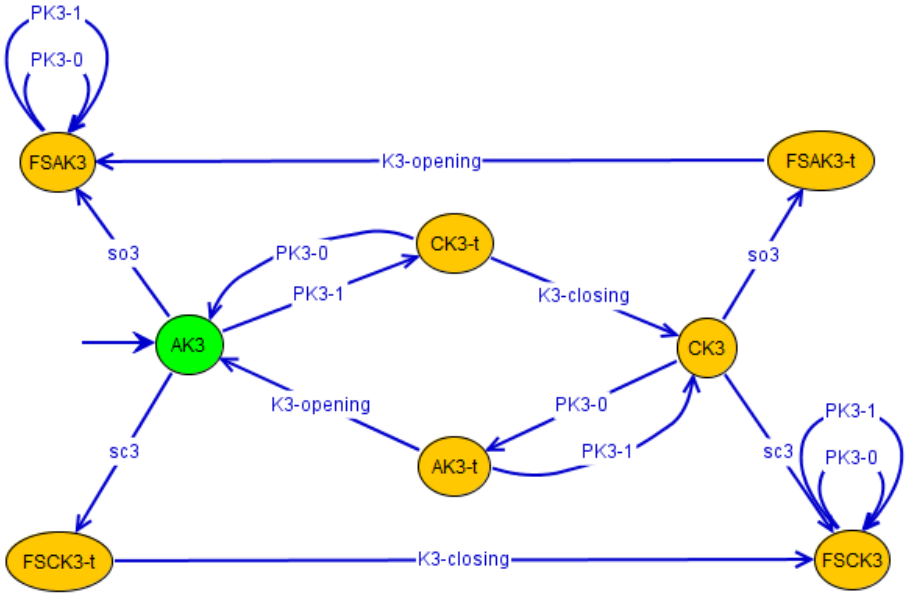


Figura 3. 16 *Modelo de FSM para el actuador 3: K3. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{AK3\}$

$X = \{AK3, CK3 - t, CK3, AK3 - t, FSAK3, FSAK3 - t, FSCK3, FSK3 - t\}$

$\Sigma_o = \{PK3 - 0, PK3 - 1\}$

$\Sigma_{uo} = \{so3, sc3, K3 - closing, K3 - opening\}$

$\Sigma_c = \{PK3 - 0, PK3 - 1\}$

$\Sigma_{uc} = \{so3, sc3, K3 - closing, K3 - opening\}$

$\Sigma_f = \{so3, sc3\}$

Los modelos de los sensores para el actuador 1 son:

Modelado del sensor SK1:

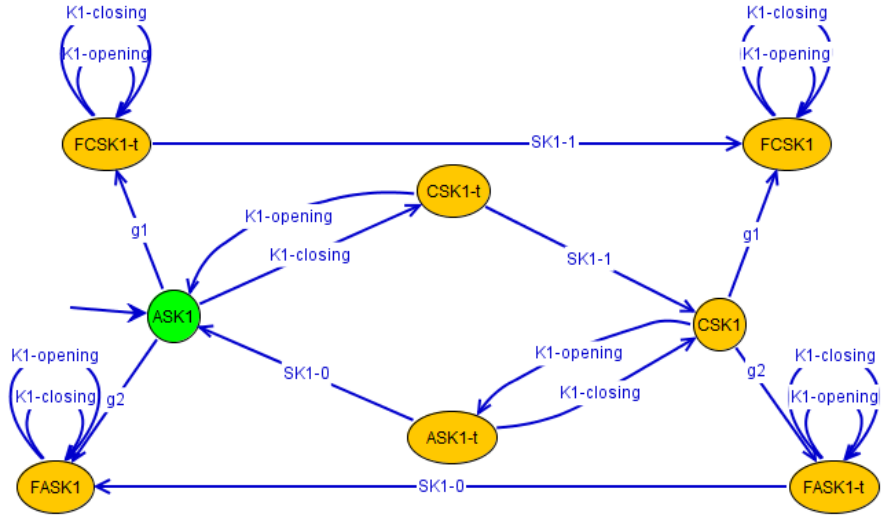


Figura 3. 17 *Modelo de FSM para el sensor SK1. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{ASK1\}$

$X = \{ASK1, CSK1 - t, CSK1, ASK1 - t, FASK1, FASK1 - t, FCSK1, FCSK1 - t\}$

$\Sigma_o = \{SK1 - 0, SK1 - 1\}$

$\Sigma_{uo} = \{g1, g2, K1 - closing, K1 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g1, g2, K1 - closing, K1 - opening\}$

$\Sigma_f = \{g1, g2\}$

Modelado del sensor SK2:

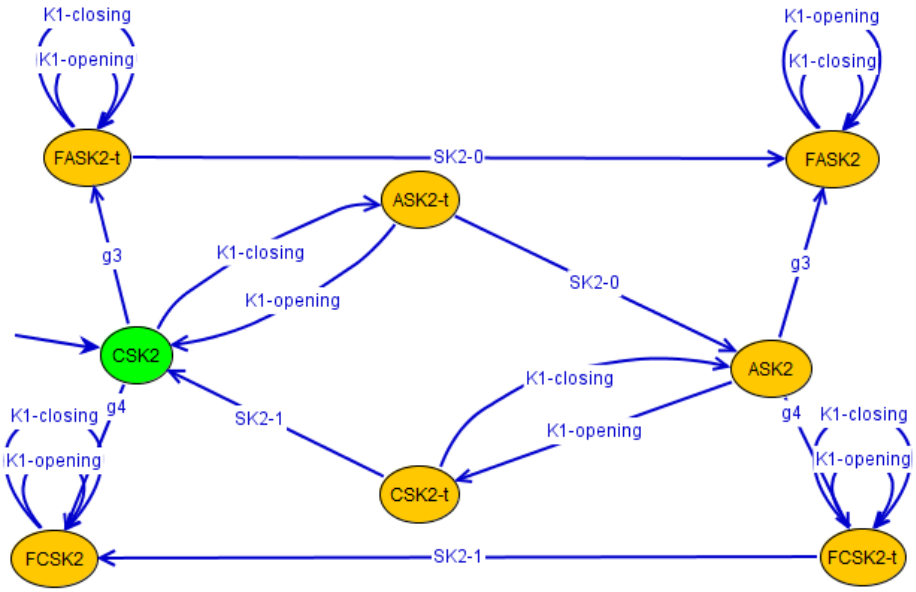


Figura 3. 18 *Modelo de FSM para el sensor SK2. Obtenido en DESUMA V3.0.*
Fuente: *Autor*

Obteniéndose:

$x_0 = \{CSK2\}$

$X = \{CSK2, CSK2 - t, ASK2, ASK2 - t, FASK2, FASK2 - t, FCSK2, FCSK2 - t\}$

$\Sigma_o = \{SK2 - 0, SK2 - 1\}$

$\Sigma_{uo} = \{g3, g4, K1 - closing, K1 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g3, g4, K1 - closing, K1 - opening\}$

$\Sigma_f = \{g3, g4\}$

Modelado del sensor SM1:

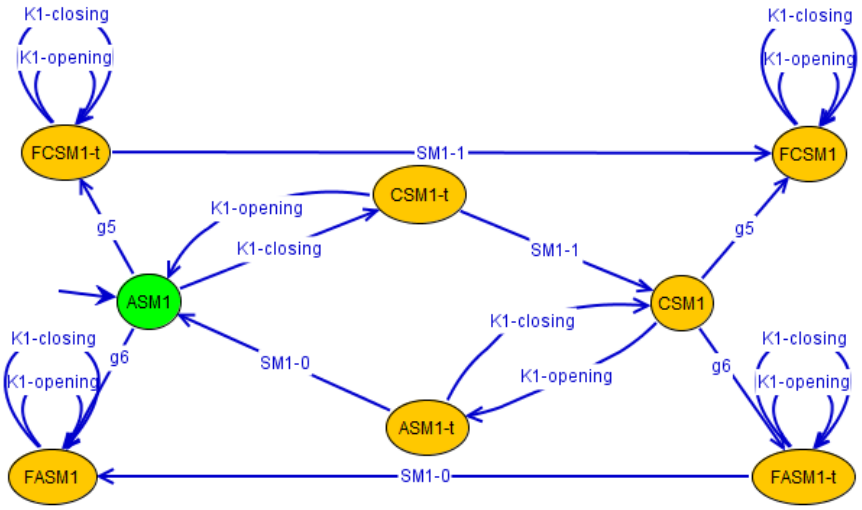


Figura 3. 19 *Modelo de FSM para el sensor SM1. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{ASM1\}$

$X = \{ASM1, ASM1-t, CSM1, CSM1-t, FASM1, FASM1-t, FCSM1, FCSM1-t\}$

$\Sigma_o = \{SM1-0, SM1-1\}$

$\Sigma_{uo} = \{g5, g6, K1-closing, K1-opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g5, g6, K1-closing, K1-opening\}$

$\Sigma_f = \{g5, g6\}$

Los modelos de los sensores para el actuador 2 son:

Modelado del sensor SK3:

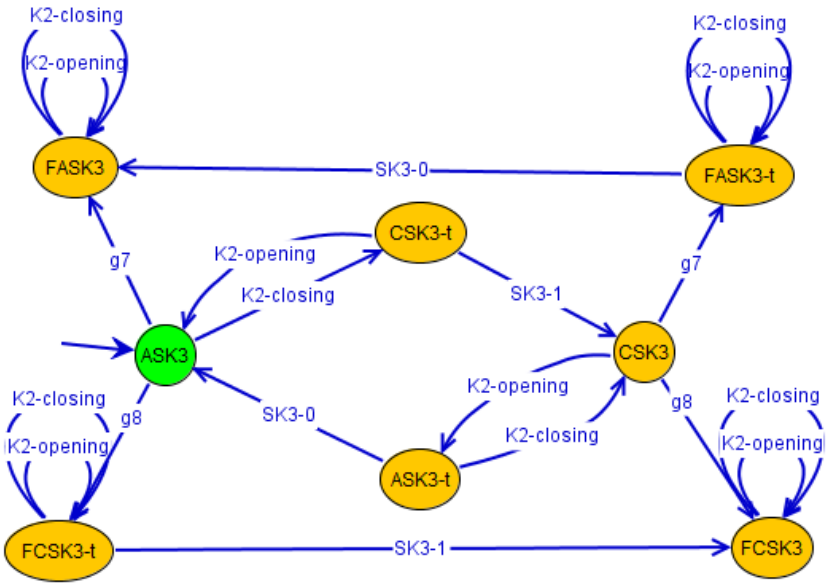


Figura 3. 20 *Modelo de FSM para el sensor SK3. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{ASK3\}$

$X = \{ASK3, ASK3 - t, CSK3, CSK3 - t, FASK3, FASK3 - t, FCSK3, FCSK3 - t\}$

$\Sigma_o = \{SK3 - 0, SK3 - 1\}$

$\Sigma_{uo} = \{g7, g8, K2 - closing, K2 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g7, g8, K2 - closing, K2 - opening\}$

$\Sigma_f = \{g7, g8\}$

Modelado del sensor SK4:

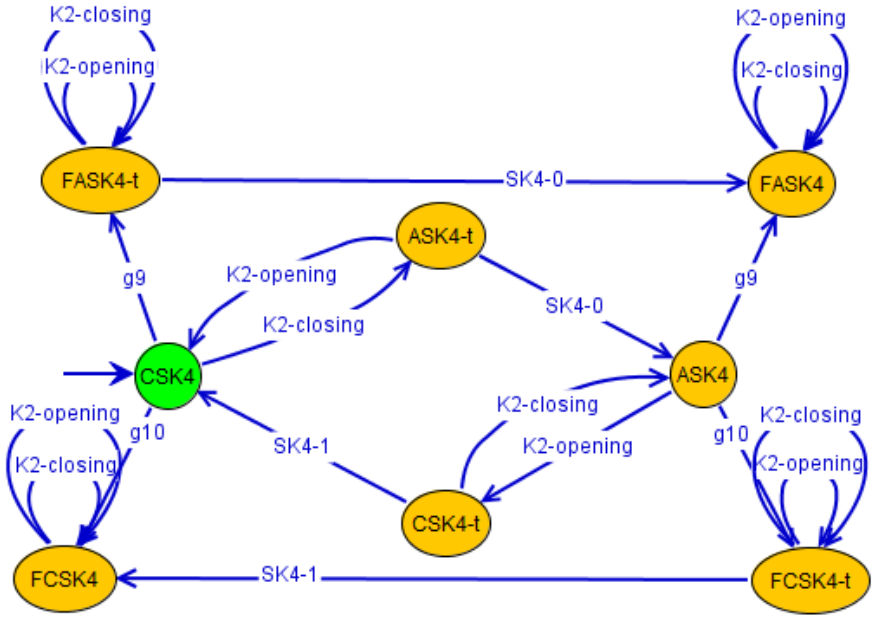


Figura 3. 21 *Modelo de FSM para el sensor SK4. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{CSK4\}$

$X = \{CSK4, CSK4 - t, ASK4, ASK4 - t, FASK4, FASK4 - t, FCSK4, FCSK4 - t\}$

$\Sigma_o = \{SK4 - 0, SK4 - 1\}$

$\Sigma_{uo} = \{g9, g10, K2 - closing, K2 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g9, g10, K2 - closing, K2 - opening\}$

$\Sigma_f = \{g9, g10\}$

Modelado del sensor SM2:

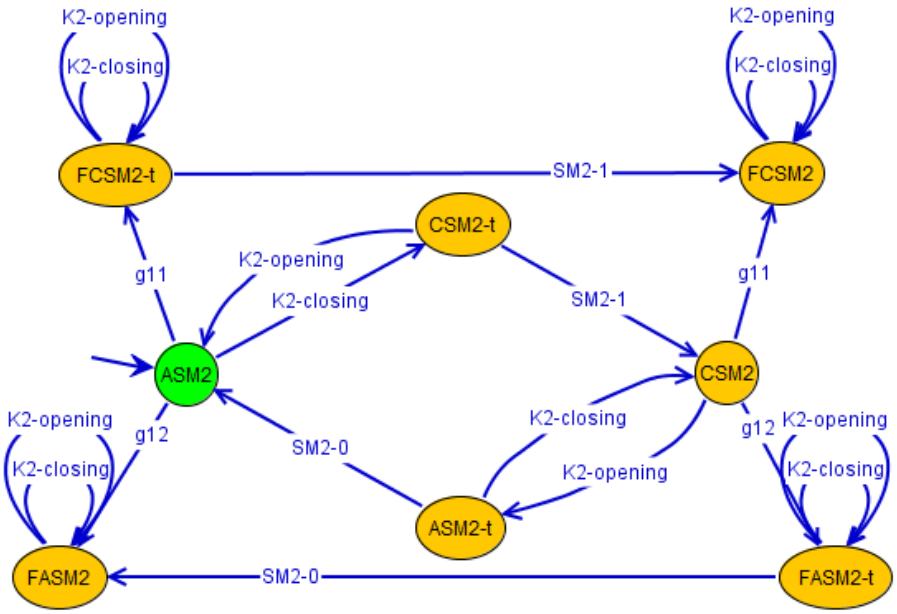


Figura 3. 22 *Modelo de FSM para el sensor SM2. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{ASM2\}$

$X = \{ASM2, ASM2 - t, CSM2, CSM2 - t, FASM2, FASM2 - t, FCSM2, FCSM2 - t\}$

$\Sigma_o = \{SM2 - 0, SM2 - 1\}$

$\Sigma_{uo} = \{g11, g12, K2 - closing, K2 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g11, g12, K2 - closing, K2 - opening\}$

$\Sigma_f = \{g11, g12\}$

Los modelos de los sensores para el actuador 3 son:

Modelado del sensor SK5:

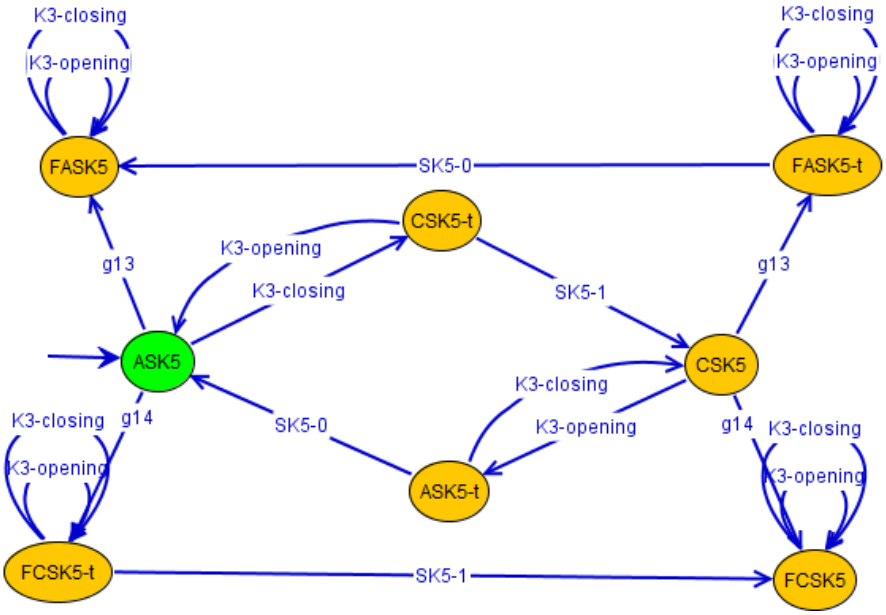


Figura 3. 23 *Modelo de FSM para el sensor SK5. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{ASK5\}$

$X = \{ASK5, ASK5-t, CSK5, CSK5-t, FASK5, FASK5-t, FCSK5, FCSK5-t\}$

$\Sigma_o = \{SK5-0, SK5-1\}$

$\Sigma_{uo} = \{g13, g14, K3-closing, K3-opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g13, g14, K3-closing, K3-opening\}$

$\Sigma_f = \{g13, g14\}$

Modelado del sensor SK6:

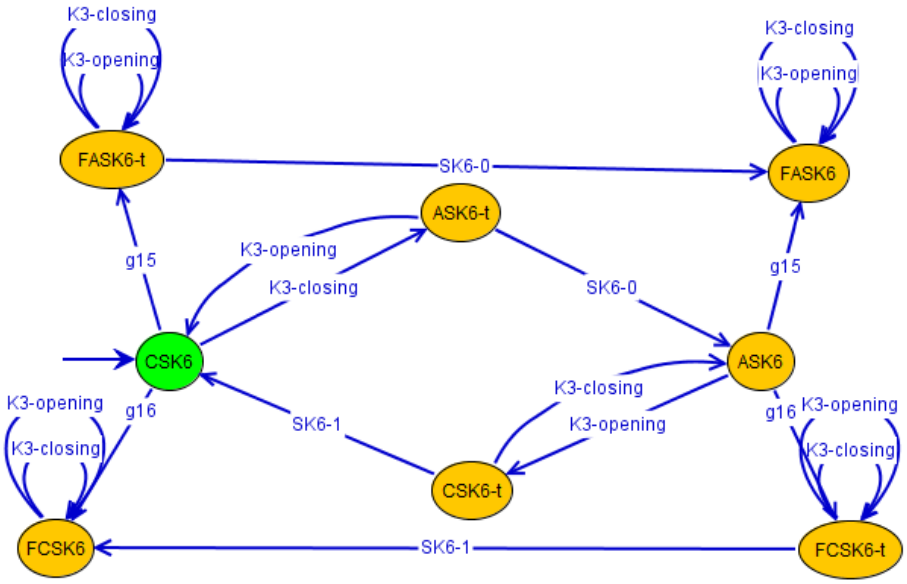


Figura 3. 24 *Modelo de FSM para el sensor SK6. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{CSK6\}$

$X = \{CSK6, ASK6 - t, ASK6, CSK6 - t, FASK6, FASK6 - t, FCSK6, FCSK6 - t\}$

$\Sigma_o = \{SK6 - 0, SK6 - 1\}$

$\Sigma_{uo} = \{g15, g16, K3 - closing, K3 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g15, g16, K3 - closing, K3 - opening\}$

$\Sigma_f = \{g15, g16\}$

Modelado del sensor SM3:

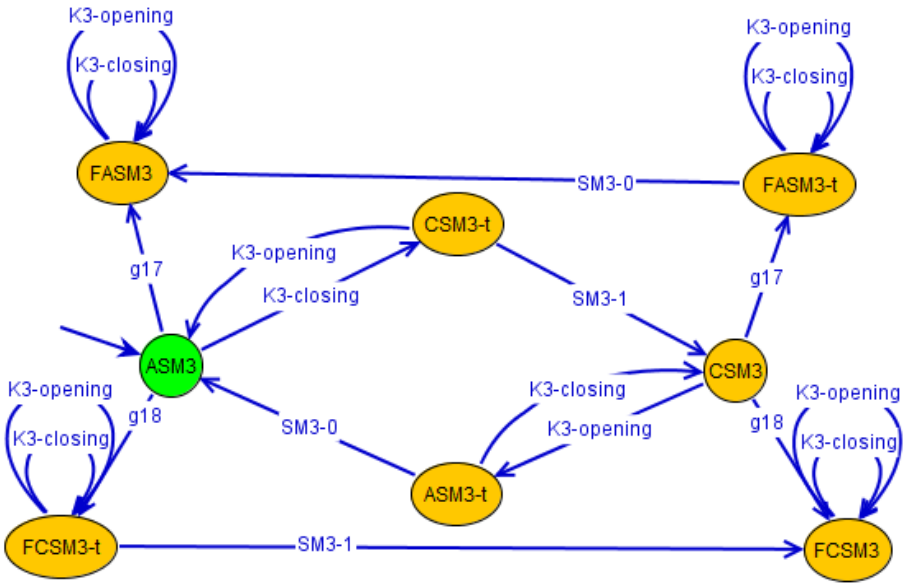


Figura 3. 25 *Modelo de FSM para el sensor SM3. Obtenido en DESUMA V3.0.*
Fuente: Autor

Obteniéndose:

$x_0 = \{ASM3\}$

$X = \{ASM3, CSM3 - t, CSM3, ASM3 - t, FASM3, FASM3 - t,\ FCSM3,\ FCSM3 - t\}$

$\Sigma_o = \{SM3 - 0, SM3 - 1\}$

$\Sigma_{uo} = \{g17, g18, K3 - closing, K3 - opening\}$

$\Sigma_c = \{\ \ \} = \emptyset$

$\Sigma_{uc} = \{g17, g18, K3 - closing, K3 - opening\}$

$\Sigma_f = \{g17, g18\}$

CAPÍTULO 4

DISEÑO DE DIAGNOSTICADORES DESCENTRALIZADOS

4.1 Generalidades

Siguiendo la metodología para establecer un sistema de diagnóstico, se debe primero plantear cual será el control que se aplicará al dispositivo modelado. (Ver los modelos del capítulo 3). Lo que llevaría a establecer secuencias que abarquen tanto el funcionamiento nominal como el extendido.

Entonces, al momento de implementar el diagnosticador siguiendo las directrices planteadas por (Sampath, 1995) en su Tesis Doctoral, en el texto presentado por (Lafortune & Cassandras, 2008), se obtiene un diagnosticador, que si bien es cierto, es capaz de "seguir" el funcionamiento del automatismo, tiene la desventaja que los estados tienden a aumentar drásticamente con la concatenación de cada uno de los dispositivos adicionados. Para el caso de estudio modelado en el capítulo 3, el modelo de la planta se obtiene a través de la composición paralela de los tres sensores y el actuador:

$$PLANTA = |K1||SK1||SK2||SM1|$$

Entonces, esta composición paralela genera *1728 estados, 7776 transiciones, 19 eventos* (solo para el actuador K1, y considerando que solo se hizo la composición paralela de los sensores y el actuador). El mismo resultado se obtiene para los otros dos actuadores, K2 y K3.

Considerando esta complejidad del autómata de la planta, la propuesta supone un diseño modular.

4.2 Planteamiento de los controladores

De esta manera, la propuesta de este trabajo es el planteamiento de un diagnosticador que agrupe el actuador con cada uno de los sensores involucrados, es

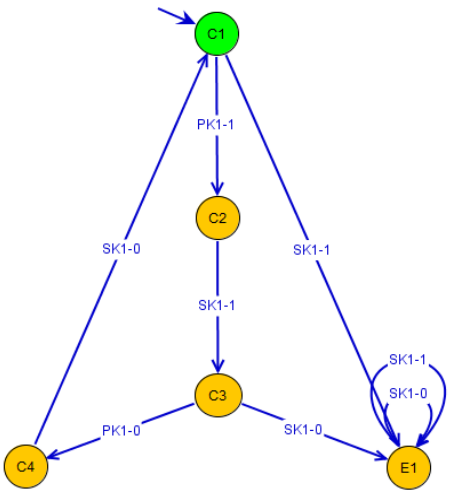
decir un esquema modular, considerando las indicaciones planteadas en (Monsalve, 2011) y (Cerrada, Ferrarini, & Dedé, 2010).

Con base en la descomposición modular planteada en la Figura 2. 10, el autómata controlador G_{ci} se lo puede obtener del controlador monolítico, sustituyendo la cadena de evento vacío ε en los eventos observables y controlables que no pertenecen a Σ^i, conjunto de eventos para cada módulo. De ahí que, el controlador para el caso modular queda definido por $G_{ci} = (X_i, \Sigma_i, \delta_i, x_0)$, donde

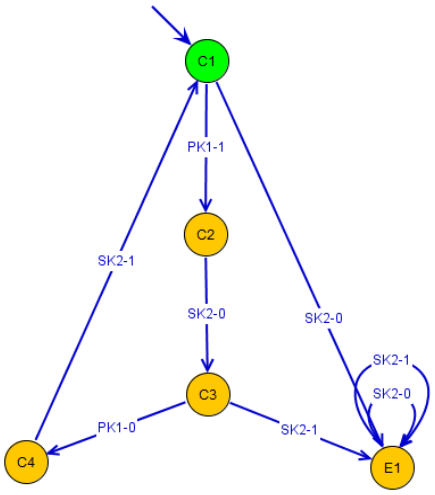
$$\delta_{ci}(x_c, \sigma) = \begin{cases} \delta_c & si\ \sigma \in \Sigma^i \\ \varepsilon & si\ \sigma \notin \Sigma^i \end{cases}$$

Para elaborar el controlador monolítico que opere sobre la planta descrita al inicio de este capítulo, se debe seguir lo descrito en (Cerrada, Ferrarini, & Dedé, 2010). Entonces, a partir de este controlador monolítico se generan los controladores individuales, considerándose que los estados C1 a C4 y E1 a E3 son posiciones que denotan estados de espera hasta que se genere la condición de transición y estado de error, respectivamente.

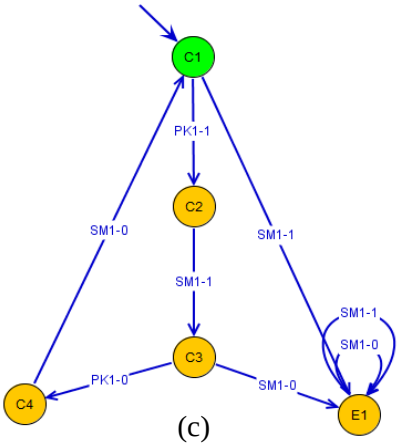
Para el subsistema (módulo) 1:



(a)



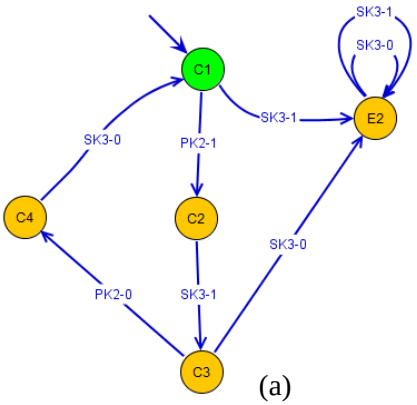
(b)



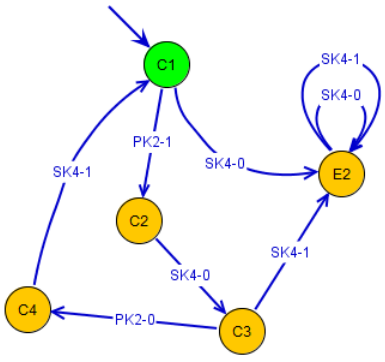
(c)

Figura 4. 1(a, b, c) *Controladores modulares para los sensores SK1, SK2 y SM1.*
Fuente: Autor

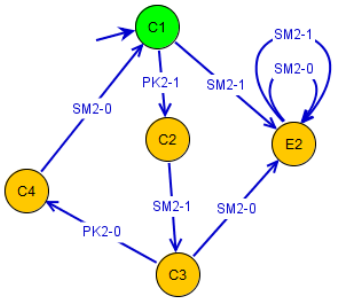
Para el subsistema (módulo) 2:



(a)



(b)



(c)

Figura 4. 2 (a, b, c) *Controladores modulares para los sensores SK3, SK4 y SM2.*
Fuente: Autor

Para el subsistema (módulo) 3:

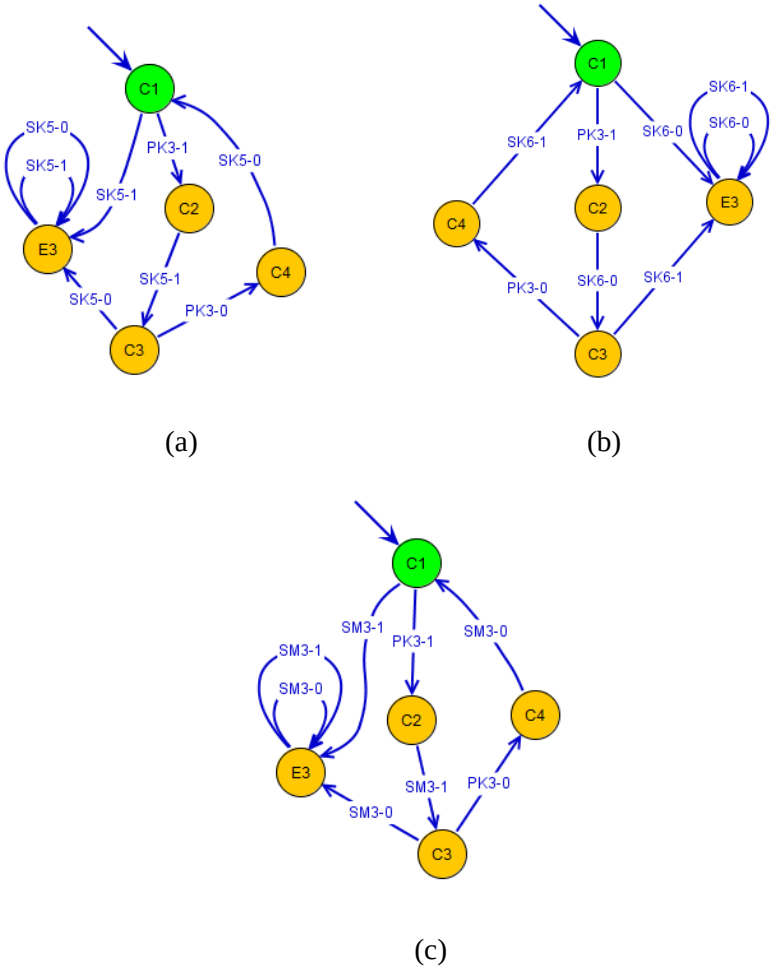


Figura 4. 3 (a, b, c) *Controladores modulares para los sensores SK5, SK6 y SM3*
Fuente: Autor

4.3 Realización de los diagnosticadores

Para definir el tipo de diagnosticador a usar por cada uno de los componentes, se debe plantear los subsistemas $Gp_i, con\ i = 1, 2, 3, \ldots$; estos se los obtiene por la composición paralela del controlador, el sensor, y el actuador común. Siendo:

Para el subsistema (módulo) 1:

$Gp_1 = |Gc_1||SK1||K1|, \quad Gp_2 = |Gc_2||SK2||K1|, \quad Gp_3 = |Gc_3||SM1||K1|,$

Para el subsistema (módulo) 2:

$Gp_4 = |Gc_4||SK3||K2|, \quad Gp_5 = |Gc_5||SK4||K2|, \quad Gp_6 = |Gc_6||SM2||K2|,$

Para el subsistema (módulo) 3:

$Gp_7 = |Gc_7||SK5||K3|, \quad Gp_8 = |Gc_8||SK6||K3|, \quad Gp_9 = |Gc_9||SM3||K3|,$

En donde cada autómata resultante Gp_i contiene *68 estados, 122 transiciones, 10 eventos.* Para demostración de su constitución se expone el Gp_1:

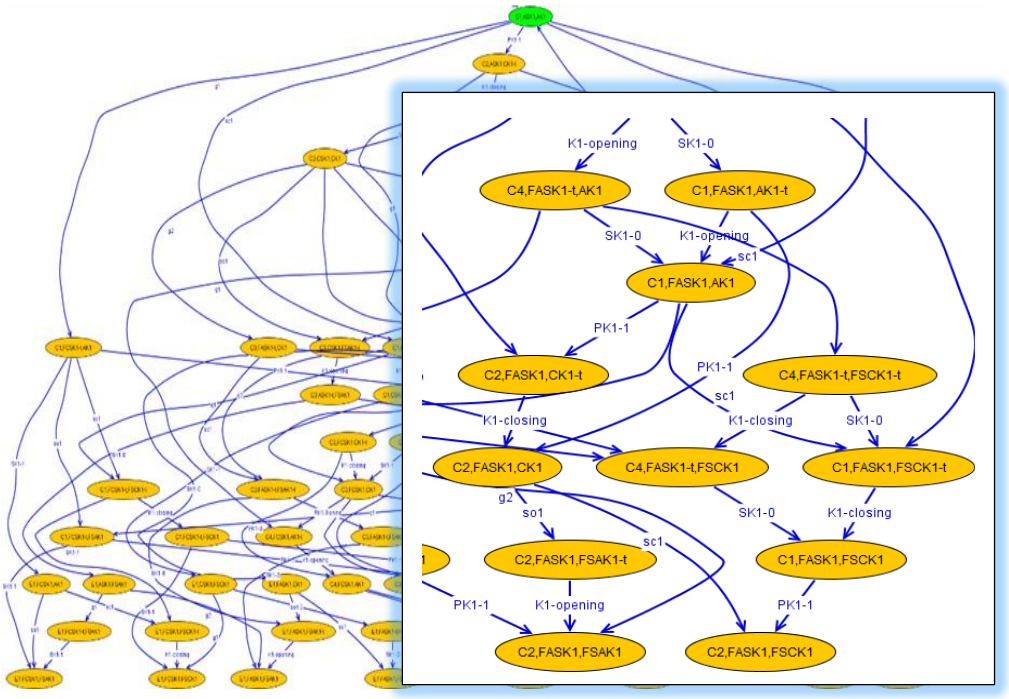


Figura 4. 4 *Autómata Gp_1, resultado de la composición paralela de Gc_1, $SK1$ y $K1$.*
Fuente: Autor

Los otros autómatas Gp_2 ... Gp_9 tienen una constitución igual, dado por el funcionamiento igual de cada sensor. Asumiendo:

- SK1, SK3, SK5 tienen igual operación, son normalmente abiertos y al pasar a cerrado, indica activación.
- SK2, SK4, SK6 tienen igual operación, son normalmente cerrados y al pasar a abierto, indica activación.
- SM1, SM2, SM3 tienen igual operación, con normalmente abiertos y al pasar a cerrado, indica activación.

Ahora, definiendo las particiones de falla (para la descripcion de cada falla referirse a la Tabla A. 2) para cada uno de los subsistemas involucrados, se tiene:

Para el subsistema (módulo) 1:

$$\sum_f{}^1 = \{g_1, g_2, so_1, sc_1\} \qquad \sum_f{}^2 = \{g_3, g_4, so_1, sc_1\} \qquad \sum_f{}^3 = \{g_5, g_6, so_1, sc_1\},$$

$$\Delta_f{}^1 = \{F_1, F_2, F_{30}, F_{31}\} \qquad \Delta_f{}^2 = \{F_3, F_4, F_{30}, F_{31}\} \qquad \Delta_f{}^3 = \{F_5, F_6, F_{30}, F_{31}\},$$

Para el subsistema (módulo) 2:

$$\sum_f{}^4 = \{g_7, g_8, so_2, sc_2\} \quad \sum_f{}^5 = \{g_9, g_{10}, so_2, sc_2\} \quad \sum_f{}^6 = \{g_{11}, g_{12}, so_2, sc_2\},$$

$$\Delta_f{}^4 = \{F_{10}, F_{11}, F_{40}, F_{41}\} \quad \Delta_f{}^5 = \{F_{12}, F_{13}, F_{40}, F_{41}\} \quad \Delta_f{}^6 = \{F_{14}, F_{15}, F_{40}, F_{41}\},$$

Para el subsistema (módulo) 3:

$$\sum_f{}^7 = \{g_{13}, g_{14}, so_3, sc_3\} \quad \sum_f{}^8 = \{g_{15}, g_{16}, so_3, sc_3\} \quad \sum_f{}^9 = \{g_{17}, g_{18}, so_3, sc_3\},$$

$$\Delta_f{}^7 = \{F_{20}, F_{21}, F_{50}, F_{51}\} \quad \Delta_f{}^8 = \{F_{22}, F_{23}, F_{50}, F_{51}\} \quad \Delta_f{}^9 = \{F_{24}, F_{25}, F_{50}, F_{51}\},$$

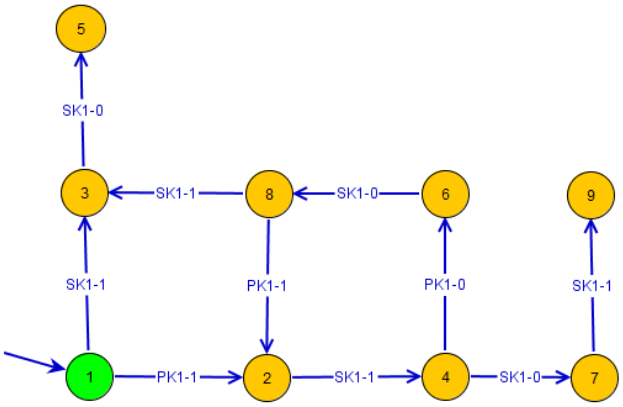
Luego, se obtienen los diagnosticadores para cada subsistema. Las fallas detectadas constituyen un conjunto de eventos no observables y no controlables, que se consideran incluidos en el conjunto de eventos no observables y no controlables del sistema.

(Debouk, 2016) indica que este conjunto de eventos se une con el conjunto de eventos observables para formar los eventos totales:

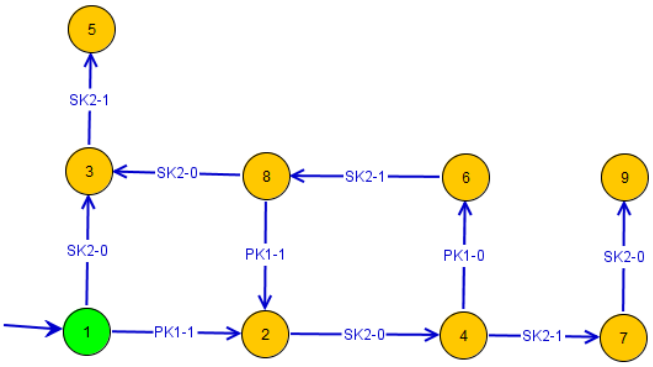
$$\Sigma = \left(\Sigma_f{}^{i} \subset \Sigma_{uo}\right) \cup \Sigma_o$$

Con esto, usando el procedimiento propuesto por (Sampath, 1995), los autómatas diagnosticadores para cada subsistema quedan de la forma:

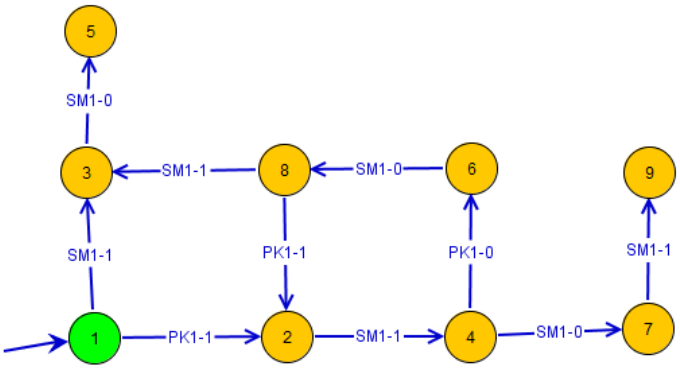
Para el subsistema (módulo) 1:



(a)



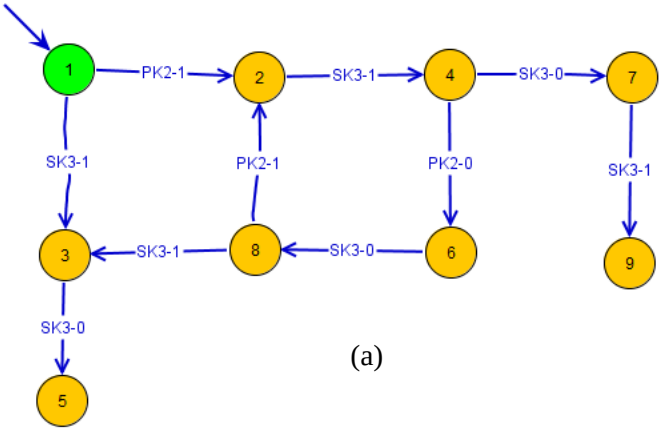
(b)



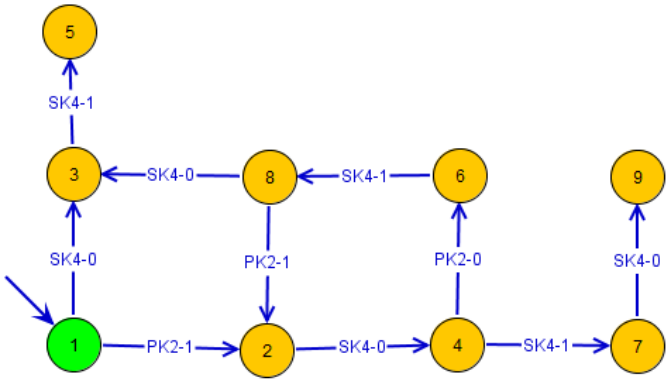
(c)

Figura 4. 5 (a, b, c) *Autómata diagnosticador* Gd_1, Gd_2, Gd_3, *para* $SK1, SK2$ *y* $SM1$
Fuente: Autor

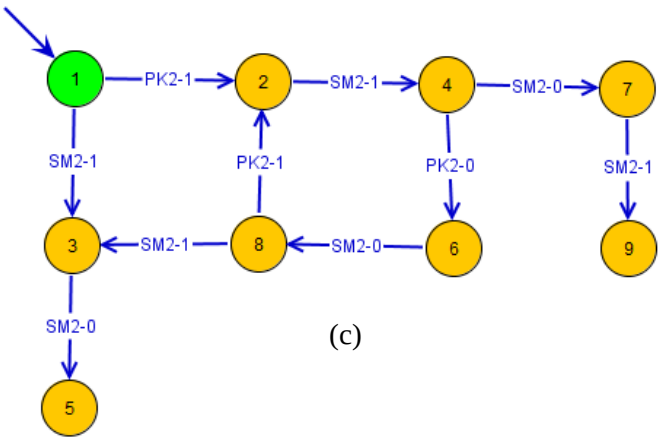
Para el subsistema (módulo) 2:



(a)



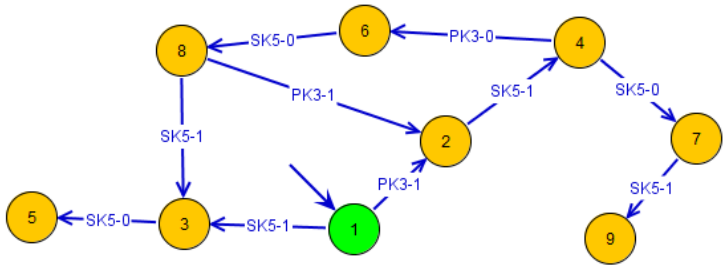
(b)



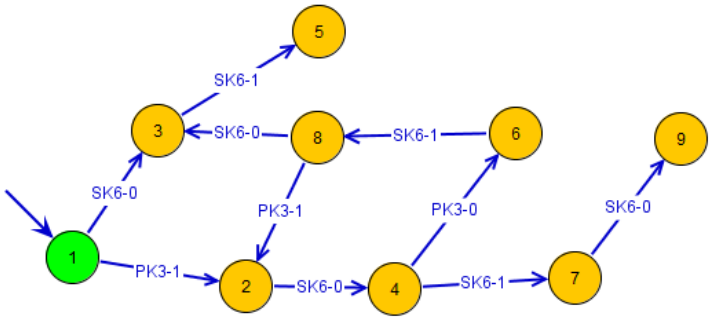
(c)

Figura 4. 6 (a, b, c) *Autómata diagnosticador* Gd_4, Gd_5, Gd_6, *para* $SK3, SK4$ *y* $SM2$
Fuente: Autor

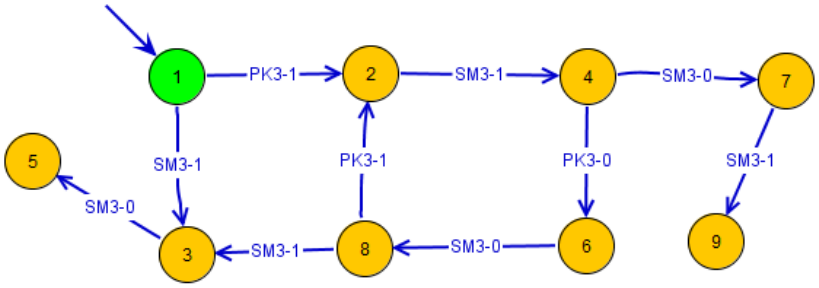
Para el subsistema (módulo) 3:



(a)



(b)



(c)

Figura 4. 7 *Autómata diagnosticador Gd_7, Gd_8, Gd_9, para $SK5, SK6$ y $SM3$*
Fuente: Autor

Cada estado del diagnosticador propone un conjunto de estados discretos posibles con las respectivas etiquetas de falla. Por ejemplo, usando el software DESUMA para el subsistema (módulo) 1, los diferentes estados estimados de la planta, por cada estado del diagnosticador, se genera en un archivo que es un texto plano, que puede verse en el Anexo 3.

Entre tanto, para el resto de los diagnosticadores, el archivo generado por el DESUMA es similar.

Aplicando el algoritmo de coordinación y protocolo de comunicación establecido en el Capítulo 2, se logra aislar la falla, según sea la secuencia de eventos observables que se generen desde los sensores.

4.4 Validación y discusión de resultados

Para efectuar la validación de lo planteado en este trabajo, se expondrá cuatro casos de estudio. Para su análisis se tomará el procedimiento descrito en (Monsalve, 2011) presentado en el Capítulo 2.

4.4.1 Caso de estudio 1

Condiciones de operación: $T_{máx}$ = 1 segundo,

1. Desde el estado inicial, se emite el comando PK1-1, y, dentro del tiempo adecuado se observa los eventos SK1-1 y SK2-0.
2. Luego, se emite el comando PK1-0, y dentro del tiempo de espera, se ve los eventos SK1-0 y SK2-1, de forma correcta.

Paso 1: determinar los subconjuntos de equilibrio actuales $q^i{}_{eq-j} \subset q^i{}_j$*, para cada diagnosticador.*

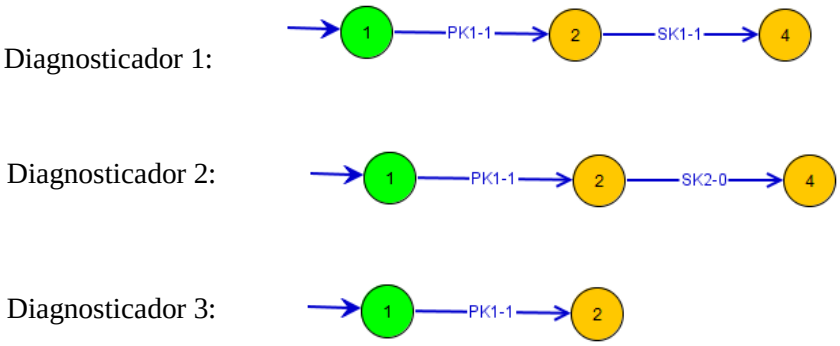


Figura 4. 8 *Traza inicial de los Autómatas diagnosticadores Gd_1, Gd_2, Gd_3, caso de estudio 1*
Fuente: Autor

Lo que genera en cada estado posible los conjuntos de eventos siguientes:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-4}$			$q^2{}_{eq-4}$			$q^3{}_{eq-2}$		
1	C3, CSK1,CK1	N	1	C3, ASK2, CK1	N	1	C2, ASM1, FSAK1	F1
2	C3, CSK1, FSCK1	F2	2	C3, ASK2, FSCK1	F2	2	C2, FASM1, CK1	F8
3	C3, FCSK1, CK1	F3	3	C3, FASK2, CK1	F5	3	C2, FASM1, FSAK1	F1F8
4	C3, FCSK1, FSAK1	F1F3	4	C3, FASK2, FSAK1	F1F5	4	C2, FASM1, FSCK1	F2F8
5	C3, FCSK1, FSCK1	F2F3	5	C3, FASK2, FSCK1	F2F5			

Tabla 4. 1 *Estados posibles de los diagnosticadores del actuador K1 (1era condición).*
Fuente: Autor

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$, determinar si F_K es F-cierta, para todo F_K que pertenece a Δ_f.

2.1. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados $q^1{}_{eq-4}$, $q^2{}_{eq-4}$ y $q^3{}_{eq-2}$. No se puede aplicar la función.

2.2. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados $q^1{}_{eq-4}$, $q^2{}_{eq-4}$ y $q^3{}_{eq-2}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.

2.3. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

- No se aplica este ítem.

Paso 3: Esperar por un nuevo conjunto de eventos observables, dentro del time-out, y para el correspondiente diagnosticador:

3.1.Determinar el estado de equilibrio sucesivo $q^i{}_{eq-j^+} \subset q^i{}_{j^+}$ *, considerando que:* $q^i{}_j$ *son los estados estimados que permiten evolucionar a la siguiente transición.*

Para los correspondientes estados estimados de $q^i{}_{eq-j^+}$*, volver a los pasos 2 y 3, respectivamente.*

Entonces, aplicando la segunda condición, los estados de equilibrio sucesivos encontrados son:

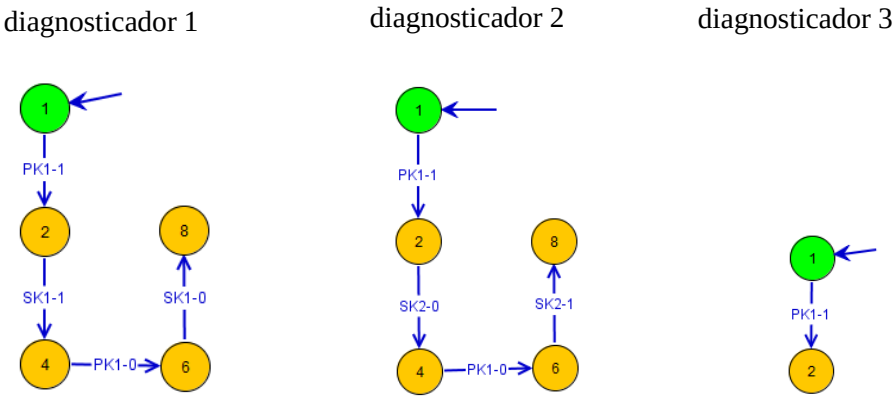


Figura 4. 9 *Traza avanzada de los Autómatas diagnosticadores* Gd_1, Gd_2, Gd_3.
Fuente: Autor

Para determinar que estados posibles son los que evolucionan, se efectúa un análisis al próximo estado de equilibrio. Siendo:

- De $q^1{}_{eq-4}$ a $q^1{}_{eq-8}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar cualquier estado (tupla) del siguiente estado. Todas son posibles.

- De la tupla 2, se puede evolucionar a la tupla 5 del siguiente estado.

- De $q^2{}_{eq-4}$ a $q^2{}_{eq-8}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar cualquier estado (tupla) del siguiente estado. Todas son posibles.
  - De la tupla 2, se puede evolucionar a la tupla 5 del siguiente estado.

- En $q^3{}_{eq-2}$:
  - No cambia las tuplas del estado posible. Dado que se queda bloqueado por el tipo de falla que se modeló.

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-4}$			$q^2{}_{eq-4}$			$q^3{}_{eq-2}$		
1	C3, CSK1,CK1	N	1	C3, ASK2, CK1	N	1	C2, ASM1, FSAK1	F1
2	C3, CSK1, FSCK1	F2	2	C3, ASK2, FSCK1	F2	2	C2, FASM1, CK1	F8
3	C3, FCSK1, CK1	F3	3	C3, FASK2, CK1	F5	3	C2, FASM1, FSAK1	F1 F8
4	C3, FCSK1, FSAK1	F1 F3	4	C3, FASK2, FSAK1	F1 F5	4	C2, FASM1, FSCK1	F2 F8
5	C3, FCSK1, FSCK1	F2 F3	5	C3, FASK2, FSCK1	F2 F5			

$q^1{}_{eq-8}$			$q^2{}_{eq-8}$		
1	C1, ASK1, AK1	N	1	C1, CSK2, AK1	N
2	C1, FASK1, AK1	F4	2	C1, FCSK2, AK1	F6
3	C1, ASK1, FSAK1	F1	3	C1, CSK2, FSAK1	F1
4	C1, FASK1, FSAK1	F1 F4	4	C1, FCSK2, FSAK1	F1 F6
5	C1, FASK1, FSCK1	F2 F4	5	C1, FCSK2, FSCK1	F2 F6

Tabla 4. 2 *Estados posibles de los diagnosticadores del actuador K1. (2da condición)*
Fuente: Autor

Regresando a los pasos 2 y 3, esto es:

Paso 2: Para cada estado estimado $q^{i}{}_{eq-j}$*, determinar si* F_K *es F-cierta, para todo* F_K *que pertenece a* Δ_f.

2.1. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados $q^{1}{}_{eq-8}$, $q^{2}{}_{eq-8}$ y $q^{3}{}_{eq-2}$. No se puede aplicar la función.

2.2. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados $q^{1}{}_{eq-8}$, $q^{2}{}_{eq-8}$ y $q^{3}{}_{eq-2}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.
- Ahora, sí se puede aplicar la regla de coordinación (iii), en los estados anteriores ($q^{1}{}_{eq-4}$, $q^{2}{}_{eq-4}$) que permitieron evolucionar a los posteriores ($q^{1}{}_{eq-8}$, $q^{2}{}_{eq-8}$). Pero se debe de considerar los estados que permiten evolucionar, y los estados que coincidan con el elemento común en el sensor bloqueado.

Resultando:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^{1}{}_{eq-4}$			$q^{2}{}_{eq-4}$			$q^{3}{}_{eq-2}$		
1	C3, CSK1,CK1	N	1	C3, ASK2, CK1	N	2	C2, FASM1, CK1	F8
2	C3, CSK1, FSCK1	F2	2	C3, ASK2, FSCK1	F2	4	C2, FASM1, FSCK1	F2F8

Tabla 4. 3 *Estados posibles, diagnosticadores del actuador K1, evolucionados*
Fuente: Autor

Al efectuar la función *merge* se produce:

tuplas	mezcla	fallas
112	C3, CSK1, CK1, ASK2, C2, FASM1	N N F8
224	C3, CSK1, FSCK1, ASK2, C2, FASM1	F2 F8

Tabla 4. 4 *Merge de los estados de la Tabla 4.3*
Fuente: Autor

Como se ve en la Tabla 4. 4, luego de realizado el *merge*, se tiene que la $\boldsymbol{F_K - cierta}$ corresponde a la **F8**.

2.1. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

Se transmite la falla cierta F8 al coordinador.

4.4.2 Caso de estudio 2

Condiciones de operación: $T_{máx}$ = 1 segundo,

1. Desde el estado inicial, se ejecuta el comando PK1-1, y, dentro del tiempo de espera, se observa el evento SM1-1, de forma correcta.
2. Luego, se emite el comando PK1-0, en el tiempo de espera permitido se observa el evento SM1-0 correctamente. No se registra eventos en SK1 y SK2.

Paso 1: Los subconjuntos de equilibrio actuales $q^i{}_{eq-j} \subset q^i{}_j$ *son:*

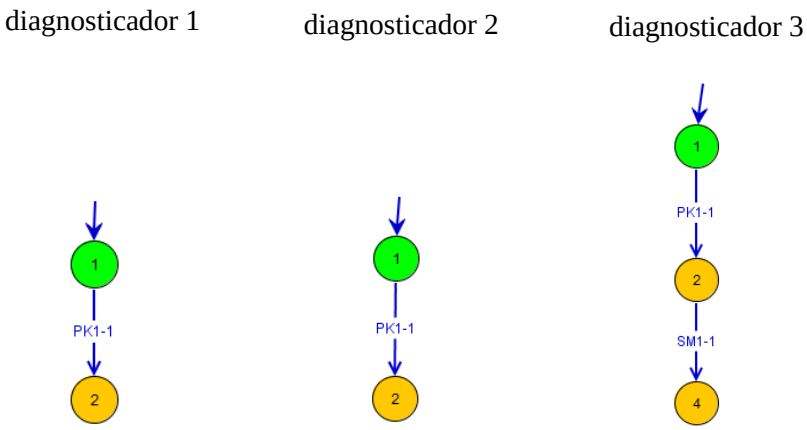


Figura 4. 10 *Traza inicial de los Autómatas diagnosticadores* Gd_1, Gd_2, Gd_3*, caso de estudio 2*
Fuente: Autor

Lo que genera en cada estado posible los conjuntos de eventos siguientes:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-2}$			$q^2{}_{eq-2}$			$q^3{}_{eq-4}$		
1	C2, ASK1, FSAK1	F1	1	C2, CSK2, FSAK1	F1	1	C3, CSM1, CK1	N
2	C2, FASK1, CK1	F4	2	C2, FCSK2, CK1	F6	2	C3, CSM1, FSCK1	F2
3	C2, FASK1, FSAK1	F1F4	3	C2, FCSK2, FSAK1	F1F6	3	C3, FCSM1, CK1	F7
4	C2, FASK1, FSCK1	F2F4	4	C2, FCSK2, FSCK1	F2F6	4	C3, FCSM1, FSAK1	F1F7
						5	C3, FCSM1, FSCK1	F2F7

Tabla 4. 5 *Estados posibles, diagnosticadores del actuador K1. (1era condición, caso de estudio 2)*
Fuente: Autor

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$, determinar si F_K es F-cierta, para todo F_K que pertenece a Δ_f.

2.1. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados $q^1{}_{eq-2}$, $q^2{}_{eq-2}$ y $q^3{}_{eq-4}$. No se puede aplicar la función.

2.2. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados $q^1{}_{eq-2}$, $q^2{}_{eq-2}$ y $q^3{}_{eq-4}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.

2.3. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

- No se aplica este ítem.

Paso 3: Esperar por un nuevo conjunto de eventos observables, dentro del time-out, y para el correspondiente diagnosticador:

3.1.Determinar el estado de equilibrio sucesivo $q^i{}_{eq-j^+} \subset q^i{}_{j^+}$, considerando que: $q^i{}_j$ son los estados estimados que permiten evolucionar a la siguiente transición.

Para los correspondientes estados estimados de $q^{i}{}_{eq-j^{+}}$*, volver a los pasos 2 y 3, respectivamente.*

Entonces, aplicando la segunda condición, los estados de equilibrio sucesivos encontrados son:

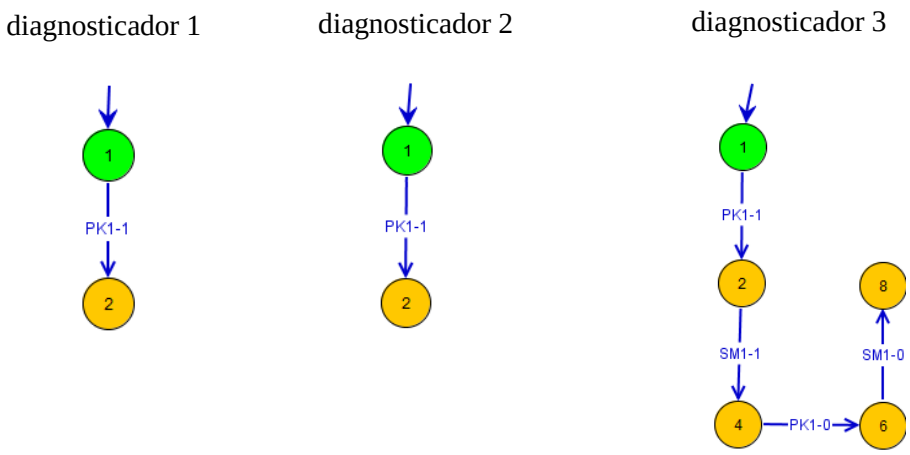


Figura 4. 11 *Traza avanzada de los Autómatas diagnosticadores* Gd_1, Gd_2, Gd_3*, Caso de estudio 2*
Fuente: Autor

Para determinar que estados posibles son los que evolucionan, se efectúa un análisis al próximo estado de equilibrio. Siendo sólo en el diagnosticador 3:

- De $q^{1}{}_{eq-2}$:
  - No se produce alteración de los estados posibles, dada el tipo de falla modelada.
- De $q^{2}{}_{eq-2}$:
  - No se produce alteración de los estados posibles, dada el tipo de falla modelada.
- De $q^{3}{}_{eq-4}$ a $q^{3}{}_{eq-8}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar cualquier estado (tupla) del siguiente estado. Todas son posibles.
  - De la tupla 2, se puede evolucionar a la tupla 5 del siguiente estado.

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-2}$			$q^2{}_{eq-2}$			$q^3{}_{eq-4}$		
1	C2, ASK1, FSAK1	F1	1	C2, CSK2, FSAK1	F1	1	C3, CSM1, CK1	N
2	C2, FASK1, CK1	F4	2	C2, FCSK2, CK1	F6	2	C3, CSM1, FSCK1	F2
3	C2, FASK1, FSAK1	F1F4	3	C2, FCSK2, FSAK1	F1F6	3	C3, FCSM1, CK1	F7
4	C2, FASK1, FSCK1	F2F4	4	C2, FCSK2, FSCK1	F2F6	4	C3, FCSM1, FSAK1	F1F7
						5	C3, FCSM1, FSCK1	F2F7

$q^3{}_{eq-8}$		
1	C1, ASM1, AK1	N
2	C1, FASM1, AK1	F8
3	C1, ASM1, FSAK1	F1
4	C1, FASM1, FSAK1	F1F8
5	C1, FASM1, FSCK1	F2F8

Tabla 4. 6 *Estados posibles de los diagnosticadores del actuador K1. (2da condición)*
Fuente: Autor

Regresando a los pasos 2 y 3, esto es:

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$, determinar si F_K es F-cierta, para todo F_K que pertenece a Δ_f.

2.1. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados $q^1{}_{eq-2}$, $q^2{}_{eq-2}$ y $q^3{}_{eq-8}$. No se puede aplicar la función.

2.2. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados $q^1{}_{eq-2}$, $q^2{}_{eq-2}$ y $q^3{}_{eq-8}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.
- Ahora, sí se puede aplicar la regla de coordinación (iii), en el estado anterior ($q^3{}_{eq-4}$) que permite evolucionar al posterior ($q^3{}_{eq-8}$). Pero

se debe de considerar el estado que permite evolucionar, y los estados que coincidan con el elemento común en los sensores bloqueados.

Resultando:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
q^1_{eq-2}			q^2_{eq-2}			q^3_{eq-4}		
2	C2, FASK1, CK1	F4	2	C2, FCSK2, CK1	F6	1	C3, CSM1, CK1	N
4	C2, FASK1, FSCK1	F2F4	4	C2, FCSK2, FSCK1	F2F6	2	C3, CSM1, FSCK1	F2

Tabla 4. 7 *Estados posibles, diagnosticadores del actuador K1, evolucionados*
Fuente: Autor

Al efectuar la función *merge* se produce:

tuplas	mezcla	fallas
221	C2, FASK1, CK1, FCSK2, C3, CSM1	F4 F6 N
442	C2, FASK1, FSCK1, FCSK2, C3, CSM1	F2 F4 F6

Tabla 4. 8 *Merge de los estados de la Tabla 4.7*
Fuente: Autor

Como se ve en la Tabla 4.8, luego de realizado el *merge*, se tiene que la $\boldsymbol{F_K - cierta}$ corresponde a la **F4F6**.

2.1. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

Se transmite la falla cierta F4F6 al coordinador.

4.4.3 Caso de estudio 3

Condiciones de operación: $T_{máx}$ = 1 segundo,

1. Estando en el estado 4 en cada uno de los diagnosticadores, se emite la orden de control PK1-0, y, dentro del tiempo máximo permitido, se ven los eventos SK1-0 y SM1-0, en forma correcta. Y, a K1 controlando su carga.
2. Se da el comando PK1-1, y se observan los eventos SK1-1 y SM1-1, correctamente.

Paso 1: Los subconjuntos de equilibrio actuales $q^i{}_{eq-j} \subset q^i{}_j$ *son*

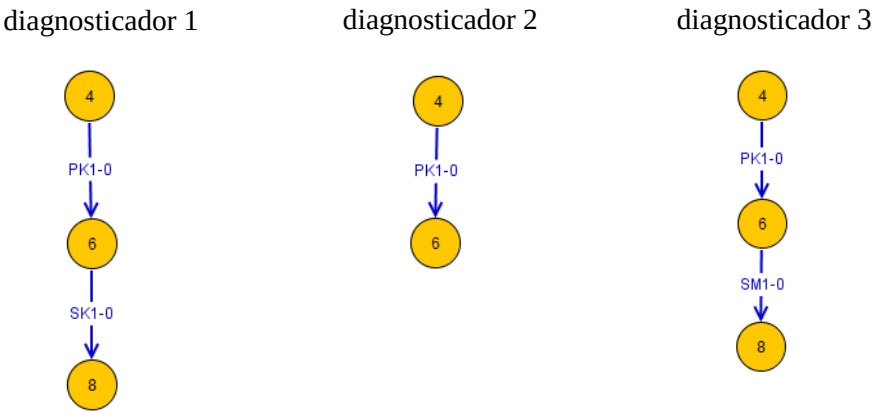


Figura 4. 12 *Traza inicial de los Autómatas diagnosticadores* Gd_1, Gd_2, Gd_3*, caso de estudio 3*
Fuente: Autor

Lo que genera en cada estado posible los conjuntos de eventos siguientes:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-8}$			$q^2{}_{eq-6}$			$q^3{}_{eq-8}$		
1	C1, ASK1, AK1	N	1	C4, ASK2, FSCK1	F2	1	C1, ASM1, AK1	N
2	C1, FASK1, AK1	F4	2	C4, FASK2, AK1	F5	2	C1, FASM1, AK1	F8
3	C1, ASK1, FSAK1	F1	3	C4, FASK2, FSAK1	F1F5	3	C1, ASM1, FSAK1	F1
4	C1, FASK1, FSAK1	F1F4	4	C4, FASK2, FSCK1	F2F5	4	C1, FASM1, FSAK1	F1F8
5	C1, FASK1, FSCK1	F2F4				5	C1, FASM1, FSCK1	F2F8

Tabla 4. 9 *Estados posibles, diagnosticadores del actuador K1. (1era condición, caso de estudio 3)*
Fuente: Autor

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$*, determinar si* F_K *es F-cierta, para todo* F_K *que pertenece a* Δ_f.

2.1. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados ${q^1}_{eq-8}$, ${q^2}_{eq-6}$ y ${q^3}_{eq-8}$. No se puede aplicar la función.

2.2. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados ${q^1}_{eq-8}$, ${q^2}_{eq-6}$ y ${q^3}_{eq-8}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.

2.3. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

- No se aplica este ítem.

Paso 3: Esperar por un nuevo conjunto de eventos observables, dentro del time-out, y para el correspondiente diagnosticador:

3.1.Determinar el estado de equilibrio sucesivo ${q^i}_{eq-j^+} \subset {q^i}_{j^+}$ *, considerando que:* ${q^i}_j$ *son los estados estimados que permiten evolucionar a la siguiente transición.*

Para los correspondientes estados estimados de ${q^i}_{eq-j^+}$*, volver a los pasos 2 y 3, respectivamente.*

Entonces, aplicando la segunda condición, los estados de equilibrio sucesivos encontrados son:

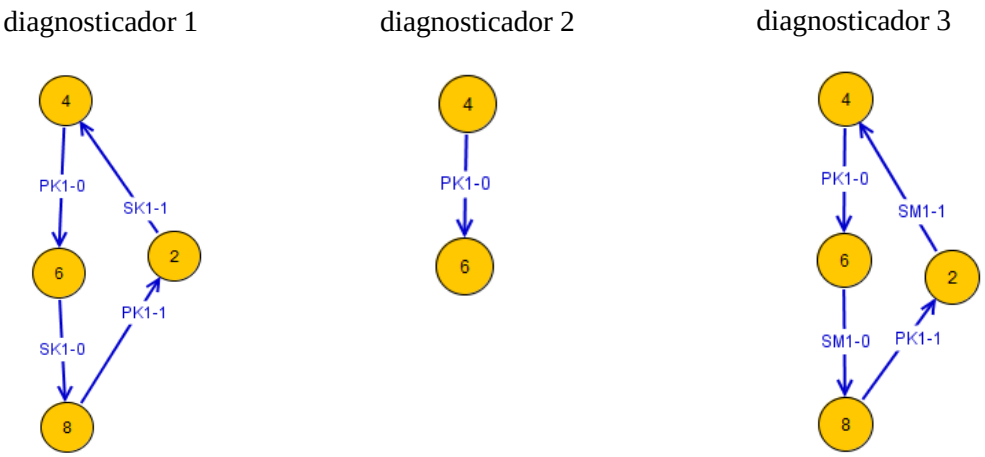


Figura 4. 13 *Traza avanzada de los Autómatas diagnosticadores* Gd_1, Gd_2, Gd_3*, Caso de estudio 3*
Fuente: Autor

Para determinar que estados posibles son los que evolucionan, se efectúa un análisis al próximo estado de equilibrio. Siendo:

- De $q^1{}_{eq-8}$ a $q^1{}_{eq-4}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar a cualquier combinación (tupla) del siguiente estado. Todas son posibles.
  - De la tupla 3, se puede evolucionar a la tupla 4 del siguiente estado.
- De $q^2{}_{eq-6}$:
  - No se produce alteración de los estados posibles, dada el tipo de falla modelada.
- De $q^3{}_{eq-8}$ a $q^3{}_{eq-4}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar a cualquier combinación (tupla) del siguiente estado. Todas son posibles.
  - De la tupla 3, se puede evolucionar a la tupla 4 del siguiente estado.

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-8}$			$q^2{}_{eq-6}$			$q^3{}_{eq-8}$		
1	C1, ASK1, AK1	N	1	C4, ASK2, FSCK1	F2	1	C1, ASM1, AK1	N
2	C1, FASK1, AK1	F4	2	C4, FASK2, AK1	F5	2	C1, FASM1, AK1	F8
3	C1, ASK1, FSAK1	F1	3	C4, FASK2, FSAK1	F1F5	3	C1, ASM1, FSAK1	F1
4	C1, FASK1, FSAK1	F1F4	4	C4, FASK2, FSCK1	F2F5	4	C1, FASM1, FSAK1	F1F8
5	C1, FASK1, FSCK1	F2F4				5	C1, FASM1, FSCK1	F2F8

$q^1{}_{eq-4}$		
1	C3, CSK1, CK1	N
2	C3, CSK1, FSCK1	F2
3	C3, FCSK1, CK1	F3
4	C3, FCSK1, FSAK1	F1F3
5	C3, FCSK1, FSCK1	F2F3

$q^3{}_{eq-4}$		
1	C3, CSM1, CK1	N
2	C3, CSM1, FSCK1	F2
3	C3, FCSM1, CK1	F7
4	C3, FCSM1, FSAK1	F1F7
5	C3, FCSM1, FSCK1	F2F7

Tabla 4. 10 *Estados posibles de los diagnosticadores del actuador K1. (2da condición)*
Fuente: Autor

Regresando a los pasos 2 y 3, esto es:

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$, determinar si F_K es F-cierta, para todo F_K que pertenece a Δ_f.

2.3. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados $q^1{}_{eq-4}$, $q^2{}_{eq-6}$ y $q^3{}_{eq-4}$. No se puede aplicar la función.

2.4. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados $q^1{}_{eq-4}$, $q^2{}_{eq-6}$ y $q^3{}_{eq-4}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.
- Ahora, sí se puede aplicar la regla de coordinación (iii), en los estados anteriores ($q^1{}_{eq-8}$, $q^3{}_{eq-8}$) que permite evolucionar a los estados posteriores ($q^1{}_{eq-4}$, $q^3{}_{eq-4}$). Pero se debe de considerar el estado que

permite evolucionar, y los estados que coincidan con el elemento común en los sensores bloqueados.

Resultando:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-8}$			$q^2{}_{eq-6}$			$q^3{}_{eq-8}$		
1	C1, ASK1, AK1	N	2	C4, FASK2, AK1	F5	1	C1, ASM1, AK1	N
3	C1, ASK1, FSAK1	F1	3	C4, FASK2, FSAK1	F1F5	3	C1, ASM1, FSAK1	F1

Tabla 4. 11 *Estados posibles, diagnosticadores del actuador K1, evolucionados*
Fuente: Autor

Al efectuar la función *merge* se produce:

tuplas	mezcla	fallas
121	C1, ASK1, AK1, C4, FASK2,ASM1	N F5 N
333	C1, ASK1, FSAK1, C4, FASK2, ASM1	F1 F5

Tabla 4. 12 *Merge de los estados de la Tabla 4.11*
Fuente: Autor

Como se ve en la Tabla 4.12, luego de realizado el *merge*, se tiene que la $\boldsymbol{F_K - cierta}$ corresponde a la **F5**.

2.1. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

Se transmite la falla cierta F5 al coordinador.

4.4.4 Caso de estudio 4

Condiciones de operación: $T_{máx}$ = 1 segundo,

1. Estando en el estado 4 en cada uno de los diagnosticadores, se emite la orden de control PK1-0, y, dentro del tiempo máximo permitido, se ven los eventos SK1-0 y SM1-0, en forma correcta. Y, a K1 controlando su carga.

2. Se da el comando PK1-1, y se observan los eventos SK1-1 y SM1-1, correctamente.

Paso 1: Los subconjuntos de equilibrio actuales $q^i{}_{eq-j} \subset q^i{}_j$ *son:*

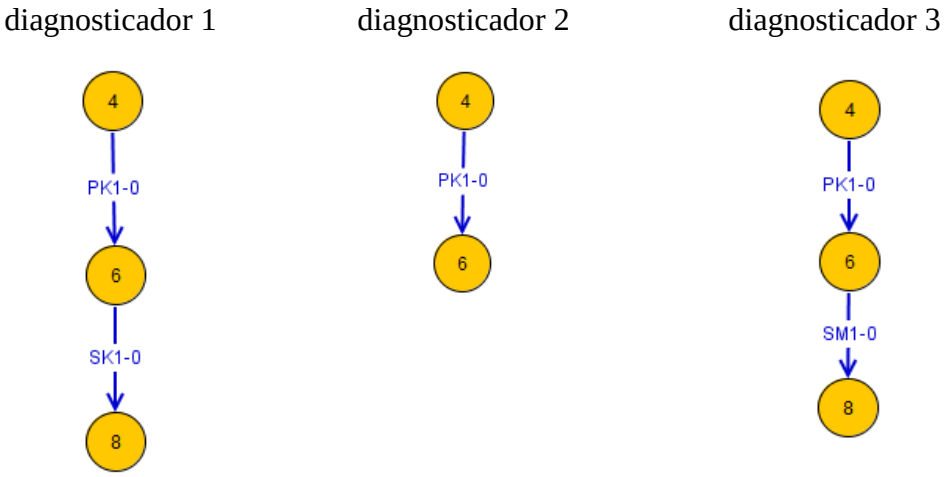


Figura 4. 14 *Traza inicial de los Autómatas diagnosticadores* Gd_1, Gd_2, Gd_3*, caso de estudio 4*
Fuente: *Autor*

Lo que genera en cada estado posible los conjuntos de eventos siguientes:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-8}$			$q^2{}_{eq-6}$			$q^3{}_{eq-8}$		
1	C1, ASK1, AK1	N	1	C4, ASK2, FSCK1	F2	1	C1, ASM1, AK1	N
2	C1, FASK1, AK1	F4	2	C4, FASK2, AK1	F5	2	C1, FASM1, AK1	F8
3	C1, ASK1, FSAK1	F1	3	C4, FASK2, FSAK1	F1F5	3	C1, ASM1, FSAK1	F1
4	C1, FASK1, FSAK1	F1F4	4	C4, FASK2, FSCK1	F2F5	4	C1, FASM1, FSAK1	F1F8
5	C1, FASK1, FSCK1	F2F4				5	C1, FASM1, FSCK1	F2F8

Tabla 4. 13 *Estados posibles, diagnosticadores del actuador K1. (1era condición, caso de estudio 4)*
Fuente: Autor

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$*, determinar si* F_K *es F-cierta, para todo* F_K *que pertenece a* Δ_f*.*

2.4. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados ${q^1}_{eq-8}$, ${q^2}_{eq-6}$ y ${q^3}_{eq-8}$. No se puede aplicar la función.

2.5. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados ${q^1}_{eq-8}$, ${q^2}_{eq-6}$ y ${q^3}_{eq-8}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.

2.6. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

- No se aplica este ítem.

Paso 3: Esperar por un nuevo conjunto de eventos observables, dentro del time-out, y para el correspondiente diagnosticador:

3.1.Determinar el estado de equilibrio sucesivo ${q^i}_{eq-j^+} \subset {q^i}_{j^+}$ *, considerando que:* ${q^i}_j$ *son los estados estimados que permiten evolucionar a la siguiente transición.*

Para los correspondientes estados estimados de ${q^i}_{eq-j^+}$*, volver a los pasos 2 y 3, respectivamente.*

Entonces, aplicando la segunda condición, los estados de equilibrio sucesivos encontrados son:

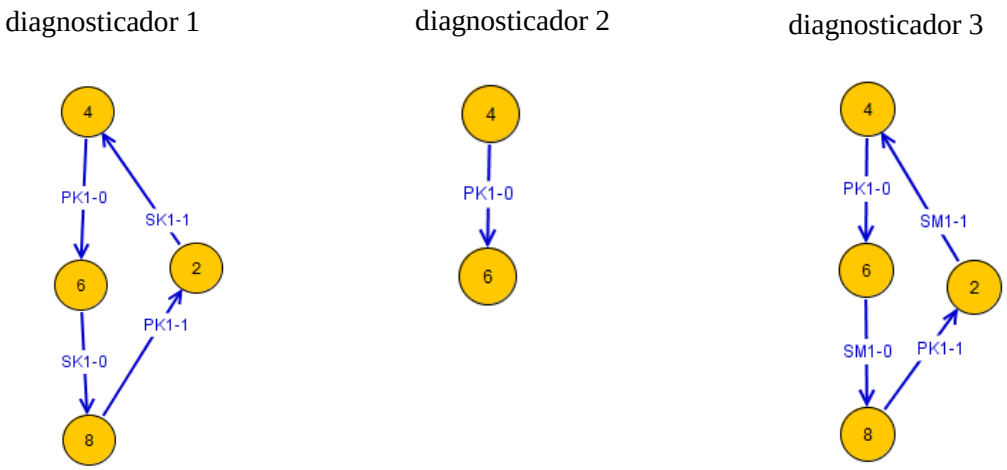


Figura 4. 15 *Traza avanzada de los Autómatas diagnosticadores Gd_1, Gd_2, Gd_3, Caso de estudio 4*
Fuente: Autor

Para determinar que estados posibles son los que evolucionan, se efectúa un análisis al próximo estado de equilibrio. Siendo:

- De $q^1{}_{eq-8}$ a $q^1{}_{eq-4}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar a cualquier combinación (tupla) del siguiente estado. Todas son posibles.
  - De la tupla 3, se puede evolucionar a la tupla 4 del siguiente estado.
- De $q^2{}_{eq-6}$:
  - No se produce alteración de los estados posibles, dada el tipo de falla modelada.
- De $q^3{}_{eq-8}$ a $q^3{}_{eq-4}$:
  - De la tupla 1, como es un estado de no-falla se puede evolucionar a cualquier combinación (tupla) del siguiente estado. Todas son posibles.
  - De la tupla 3, se puede evolucionar a la tupla 4 del siguiente estado.

diagnosticador 1			diagnosticador 2			diagnosticador 3		
$q^1{}_{eq-8}$			$q^2{}_{eq-6}$			$q^3{}_{eq-8}$		
1	C1, ASK1, AK1	N	1	C4, ASK2, FSCK1	F2	1	C1, ASM1, AK1	N
2	C1, FASK1, AK1	F4	2	C4, FASK2, AK1	F5	2	C1, FASM1, AK1	F8
3	C1, ASK1, FSAK1	F1	3	C4, FASK2, FSAK1	F1F5	3	C1, ASM1, FSAK1	F1
4	C1, FASK1, FSAK1	F1F4	4	C4, FASK2, FSCK1	F2F5	4	C1, FASM1, FSAK1	F1F8
5	C1, FASK1, FSCK1	F2F4				5	C1, FASM1, FSCK1	F2F8

$q^1{}_{eq-4}$		
1	C3, CSK1, CK1	N
2	C3, CSK1, FSCK1	F2
3	C3, FCSK1, CK1	F3
4	C3, FCSK1, FSAK1	F1F3
5	C3, FCSK1, FSCK1	F2F3

$q^3{}_{eq-4}$		
1	C3, CSM1, CK1	N
2	C3, CSM1, FSCK1	F2
3	C3, FCSM1, CK1	F7
4	C3, FCSM1, FSAK1	F1F7
5	C3, FCSM1, FSCK1	F2F7

Tabla 4. 14 *Estados posibles de los diagnosticadores del actuador K1. (2da condición)*
Fuente: Autor

Regresando a los pasos 2 y 3, esto es:

Paso 2: Para cada estado estimado $q^i{}_{eq-j}$, determinar si F_K es F-cierta, para todo F_K que pertenece a Δ_f.

2.5. Si alguna falla es $F_K - cierta$, aplicar la función *match* con el resto de los estados de equilibrio para aislar nuevas fallas, conforme a la regla de coordinación (ii).

- Dado que no existe ninguna etiqueta que se repita en los estados $q^1{}_{eq-4}$, $q^2{}_{eq-6}$ y $q^3{}_{eq-4}$. No se puede aplicar la función.

2.6. Si ninguna falla es aislada, aplicar la función *merge* y aislar nuevas fallas conforme a la regla de coordinación (ii) y (iii).

- Conforme a la regla de coordinación (ii), se aplica la función *merge* con los estados $q^1{}_{eq-4}$, $q^2{}_{eq-6}$ y $q^3{}_{eq-4}$. Pero, como todos los estados son posibles, no se logra establecer ninguna $F_K - cierta$.
- Ahora, sí se puede aplicar la regla de coordinación (iii), en los estados anteriores ($q^1{}_{eq-8}$, $q^3{}_{eq-8}$) que permite evolucionar a los estados posteriores ($q^1{}_{eq-4}$, $q^3{}_{eq-4}$). Pero se debe de considerar el estado que

permite evolucionar, y los estados que coincidan con el elemento común en los sensores bloqueados.

Resultando:

diagnosticador 1			diagnosticador 2			diagnosticador 3		
q^{1}_{eq-8}			q^{2}_{eq-6}			q^{3}_{eq-8}		
1	C1, ASK1, AK1	N	2	C4, FASK2, AK1	F5	1	C1, ASM1, AK1	N
3	C1, ASK1, FSAK1	F1	3	C4, FASK2, FSAK1	F1F5	3	C1, ASM1, FSAK1	F1

Tabla 4. 15 *Estados posibles, diagnosticadores del actuador K1, evolucionados*
Fuente: Autor

Al efectuar la función *merge* se produce:

tuplas	mezcla	fallas
121	C1, ASK1, AK1, C4, FASK2,ASM1	N F5 N
333	C1, ASK1, FSAK1, C4, FASK2, ASM1	F1 F5

Tabla 4. 16 *Merge de los estados de la Tabla 4.15*
Fuente: Autor

Como se ve en la Tabla 4.16, luego de realizado el *merge*, se tiene que la $\boldsymbol{F_K - cierta}$ corresponde a la **F5**.

2.1. Si alguna falla es detectada, transmita la información al coordinador, e ir al paso 2.1.

Se transmite la falla cierta F5 al coordinador.

En consecuencia, estos casos de estudio lo que hacen es aplicar un algoritmo que se debe seguir al producirse una falla en determinado sensor de la banda 1. Luego, si se desea identificar alguna falla en las otras dos bandas, se debe seguir la misma secuencia explicada arriba.

CAPÍTULO 5

AMBIENTE DE SIMULACIÓN

5.1 Especificaciones

Una vez determinados los casos de prueba, de manera teórica, y, establecido el procedimiento del procesamiento de los datos (*pasos a seguir para la verificación*), se procede al diseño del "ambiente de simulación", para generar un prototipo de software que pueda ser extendido a una aplicación industrial, por cuanto bastaría posteriormente definir los protocolos de comunicación adecuados entre la planta y el sistema supervisor, para hacer el seguimiento de los eventos observados, desde los sensores.

Se escoge la plataforma "Java", dado que presenta cierta versatilidad de programación, (Eckel, 2006), (Bloch, 2008), (Llobet, 2009). Y, como compilador se usará la plataforma NetBeans IDE 8.2; que es una plataforma multiuso, empleada para correr diferentes lenguajes de programación, (Bock, 2012).

5.2 Programa de simulación

Al iniciar el programa, se presenta la siguiente pantalla:

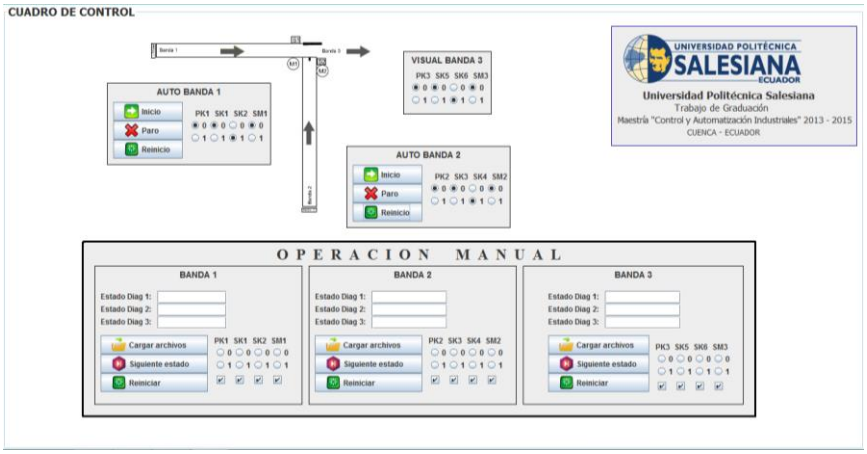


Figura 5. 1 *Pantalla principal del programa de emulación*
Fuente: Autor

5.2.1 Operación Manual

En esta fase se puede probar u operar cada sensor de cada subsistema en forma individual. Ya se para que tome una secuencia "normal", o para hacerle operar con alguna "falla inducida".

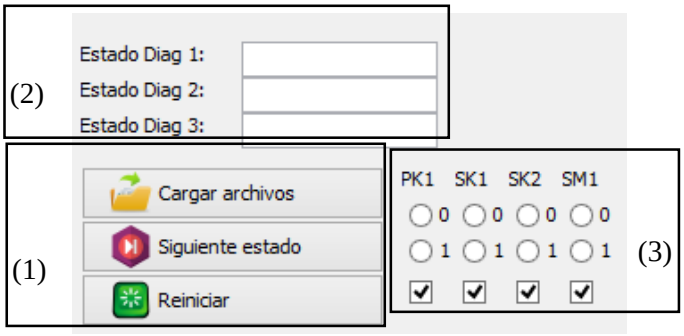


Figura 5. 2 *Parte manual de activación.*
Fuente: Autor

En las tres bandas la operación es igual; entonces, basados en la Figura 5.2:

Las acciones en los botones de (1) permiten:

Cargar archivos: permite cargar los archivos con la información de los estados estimados por los diagnosticadores locales, los cuales fueron previamente generados con el software DESUMA.

Siguiente estado: es un avance individual de cada banda, lleva al siguiente estado que le corresponde dentro del autómata diagnosticador.

Reiniciar: permite un reseteo de las variables del programa, y, lleva al inicio todos los autómatas diagnosticadores.

Los cuadros de texto, de (2), permiten visualizar el estado en que se encuentran los autómatas diagnosticadores.

Si se desea introducir alguna falla para probar el sistema, se lo puede efectuar por medio de (3); al efectuar un click en ○ para indicar si se desea una activación "1" , o una desactivación "0". En cambio, para habilitar, se hace click en ☑; si esta con el "visto" implica que el sensor en cuestión esta "habilitado" (sin falla), y sin el "visto" es que no está habilitado (con falla).

5.2.2 Operación Automática

En esta fase se puede ejecutar toda la secuencia de operación, empezando por banda 1 y banda 3, o, banda 2 y banda 3, indistintamente. Si se desea incluir una falla, se la puede colocar en (3) de la Figura 5. 2, luego se da click en Inicio y se obtiene la operación con la falla incluida. Esto se visualiza más adelante con los ejemplos planteados.

Figura 5. 3 *Parte automática de activación*
Fuente: Autor

Con base en la Figura 5.3, se tiene que:

Inicio: es el inicio de la secuencia automática. Visualizándose los estados de los autómatas diagnosticadores en (2), del cuadro de operación manual.

Paro: detiene la secuencia en cualquier momento.

Reinicio: permite volver al inicio del ambiente de simulación, para una nueva prueba.

Así mismo, se puede visualizar los estados mecánicos de los sensores, es decir, la posición en que se encuentran en cada momento.

Esto para las dos bandas de operación, pero la para tercera banda, solo se puede visualizar los estados de los sensores.

5.3 Evaluación y presentación de resultados

Para determinar el tipo de operación del ambiente de simulación, se establecerá el análisis con los dos primeros casos de estudio. Ya que los otros casos son similares en la cuestión de simulación.

5.3.1 Caso 1

Para empezar a evaluar el caso 1, se debe conocer los estados que se alcanzan en la condición 1, Figura 5. 4. Siendo:

Diagnosticador 1 → **Id = 4**

C3,CSK1,CK1 N
C3,CSK1,FSAK1-t F1
C3,CSK1,FSCK1 F2
C3,FCSK1,CK1 F3
C3,FCSK1,CK1-t F3
C3,FCSK1,FSAK1 F1F3
C3,FCSK1,FSAK1-t F1F3
C3,FCSK1,FSCK1 F2F3
C3,ASK1-t,FSAK1 F1
C3,FASK1-t,CK1 F4
C3,FASK1-t,FSAK1-t F1F4
C3,FASK1-t,FSCK1 F2F4
C3,FASK1-t,FSAK1 F1F4
Total pairs = 13
Uncertain: F1 F2 F3 F4
PK1-0 -> 6
SK1-0 -> 7

Diagnosticador 2 → **Id = 4**

C3,ASK2,CK1 N
C3,ASK2,FSAK1-t F1
C3,ASK2,FSCK1 F2
C3,FASK2,CK1 F5
C3,FASK2,CK1-t F5
C3,FASK2,FSAK1 F1F5
C3,FASK2,FSAK1-t F1F5
C3,FASK2,FSCK1 F2F5
C3,FCSK2-t,CK1 F6
C3,FCSK2-t,FSAK1-t F1F6
C3,CSK2-t,FSAK1 F1
C3,FCSK2-t,FSCK1 F2F6
C3,FCSK2-t,FSAK1 F1F6
Total pairs = 13
Uncertain: F1 F2 F5 F6
PK1-0 -> 6
SK2-1 -> 7

Diagnosticador 3 → **Id = 2**

C2,ASM1,CK1-t N
C2,ASM1,FSAK1 F1
C2,CSM1-t,FSCK1 F2
C2,FASM1,CK1-t F8
C2,FASM1,CK1 F8
C2,FASM1,FSAK1 F1F8
C2,FASM1,FSCK1 F2F8
C2,FCSM1-t,CK1-t F7
C2,FCSM1-t,FSAK1 F1F7
C2,FCSM1-t,FSCK1 F2F7
C2,CSM1-t,CK1 N
C2,CSM1-t,FSAK1-t F1
C2,FASM1,FSAK1-t F1F8
C2,FCSM1-t,CK1 F7
C2,FCSM1-t,FSAK1-t F1F7
Total pairs = 15
Uncertain: F1 F2 F7 F8
SM1-1 -> 4

Correspondientes a los estados de parada, según los diagramas de la Figura 4. 8 del capítulo 4.

Una vez que se corre el programa, se tiene los estados válidos para el análisis:

**** Combinacion Diag1: ****
C3,CSK1,CK1 N
C3,CSK1,FSCK1 F2
C3,FCSK1,CK1 F3
C3,FCSK1,FSAK1 F1F3
C3,FCSK1,FSCK1 F2F3

**** Combinacion Diag2: ****
C3,ASK2,CK1 N
C3,ASK2,FSCK1 F2
C3,FASK2,CK1 F5
C3,FASK2,FSAK1 F1F5
C3,FASK2,FSCK1 F2F5

**** Combinacion Diag3: ****

C2,ASM1,FSAK1 F1
C2,FASM1,CK1 F8
C2,FASM1,FSAK1 F1F8
C2,FASM1,FSCK1 F2F8

Entre tanto, el análisis de las funciones adecuadas produce:

****
**** Resultado de la operación Merge: ****
****
C3,C2,CSK1,ASK2,FASM1,CK1 NF8
C3,C2,CSK1,FASK2,FASM1,CK1 NF5F8
C3,C2,CSK1,ASK2,FASM1,FSCK1 F2F8
C3,C2,CSK1,FASK2,FASM1,FSCK1 F2F5F8
C3,C2,FCSK1,ASK2,FASM1,CK1 F3NF8
C3,C2,FCSK1,FASK2,FASM1,CK1 F3F5F8
C3,C2,FCSK1,FASK2,ASM1,FSAK1 F1F3F5
C3,C2,FCSK1,FASK2,FASM1,FSAK1 F1F3F5F8
C3,C2,FCSK1,ASK2,FASM1,FSCK1 F2F3F8
C3,C2,FCSK1,FASK2,FASM1,FSCK1 F2F3F5F8

De lo que se desprende que: No se ha encontrado FCierta

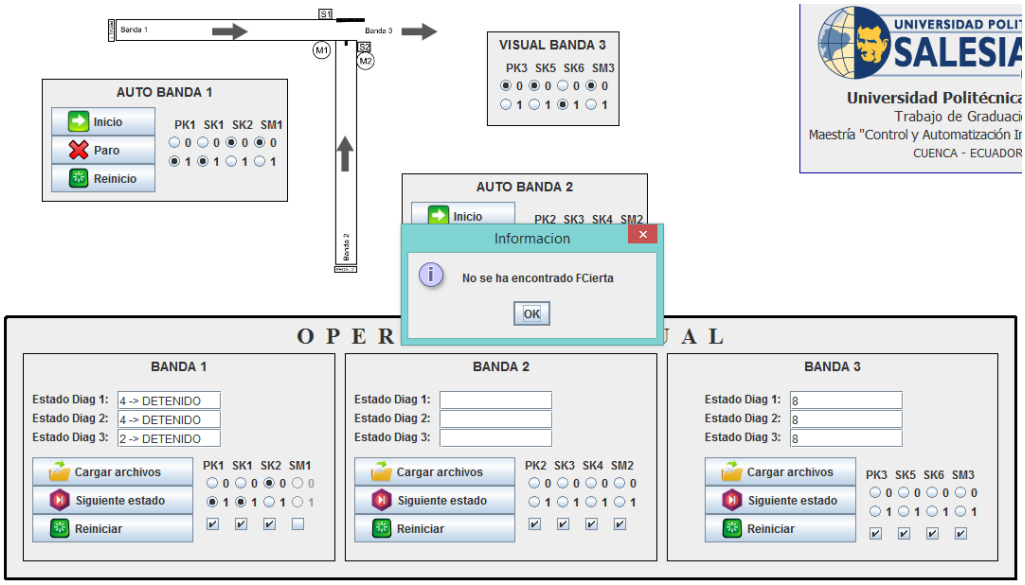


Figura 5. 4 *Resultado de investigación de la primera condición*
Fuente: Autor

Al momento de introducir la segunda condición, se logra que el algoritmo programado realice los cálculos necesarios, indicados en el capítulo 4 – caso de estudio 1, y genere el resultado demostrado anteriormente.

Alcanzando los estados de llegada, Figura 5. 5, de cada diagnosticador:

Diagnosticador 1 → **Id = 8**

C1,ASK1,AK1 N
C1,FCSK1-t,AK1 F3
C1,FASK1,AK1 F4
C1,ASK1,FSAK1 F1
C1,ASK1,FSCK1-t F2
C1,FCSK1-t,FSAK1 F1F3
C1,FASK1,FSAK1 F1F4
C1,FCSK1-t,FSCK1-t F2F3
C1,FASK1,FSCK1-t F2F4
C1,CSK1-t,FSCK1 F2
C1,FASK1,AK1-t F4
C1,FASK1,FSCK1 F2F4
C1,FCSK1-t,FSCK1 F2F3
Total pairs = 13

Uncertain: F1 F2 F3 F4
PK1-1 -> 2
SK1-1 -> 3

Diagnosticador 2 → **Id = 8**

C1,CSK2,AK1 N
C1,FASK2-t,AK1 F5
C1,FCSK2,AK1 F6
C1,CSK2,FSAK1 F1
C1,CSK2,FSCK1-t F2
C1,ASK2-t,FSCK1 F2
C1,FASK2-t,FSAK1 F1F5
C1,FCSK2,FSAK1 F1F6
C1,FASK2-t,FSCK1-t F2F5
C1,FCSK2,FSCK1-t F2F6
C1,FASK2-t,FSCK1 F2F5
C1,FCSK2,AK1-t F6
C1,FCSK2,FSCK1 F2F6
Total pairs = 13
Uncertain: F1 F2 F5 F6
PK1-1 -> 2
SK2-0 -> 3

Diagnosticador 3 → **Id = 2**

C2,ASM1,CK1-t N
C2,ASM1,FSAK1 F1
C2,CSM1-t,FSCK1 F2
C2,FASM1,CK1-t F8
C2,FASM1,CK1 F8
C2,FASM1,FSAK1 F1F8
C2,FASM1,FSCK1 F2F8
C2,FCSM1-t,CK1-t F7
C2,FCSM1-t,FSAK1 F1F7
C2,FCSM1-t,FSCK1 F2F7
C2,CSM1-t,CK1 N
C2,CSM1-t,FSAK1-t F1
C2,FASM1,FSAK1-t F1F8
C2,FCSM1-t,CK1 F7
C2,FCSM1-t,FSAK1-t F1F7
Total pairs = 15
Uncertain: F1 F2 F7 F8
SM1-1 -> 4

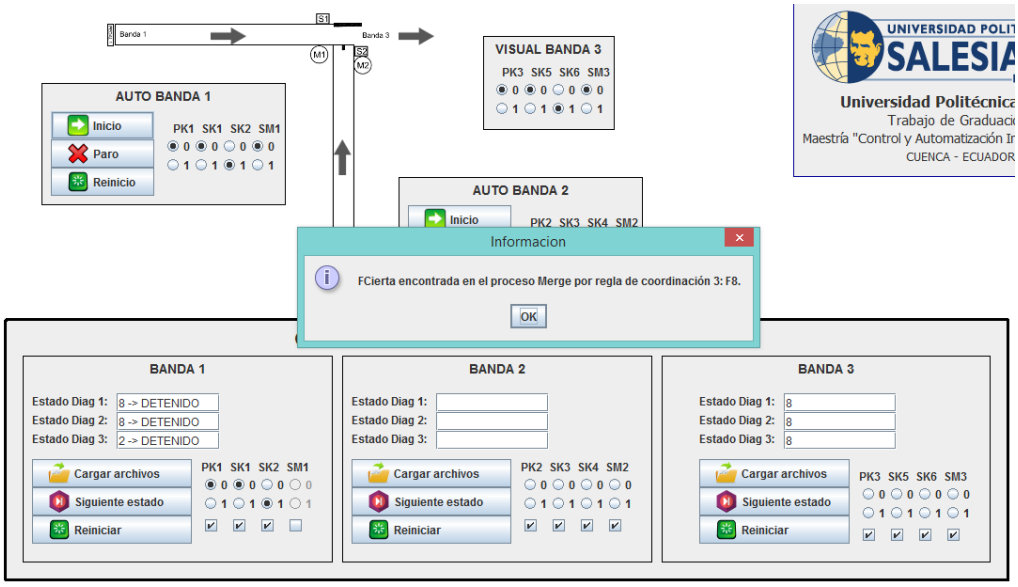


Figura 5. 5 *Resultado final, con la segunda condición*
Fuente: Autor

El programa arroja los resultados que validan la falla presentada:

**** Combinacion Diag1: ****
C1,ASK1,AK1 N
C1,FASK1,AK1 F4
C1,ASK1,FSAK1 F1
C1,FASK1,FSAK1 F1F4
C1,FASK1,FSCK1 F2F4

**** Combinacion Diag2: ****
C1,CSK2,AK1 N
C1,FCSK2,AK1 F6
C1,CSK2,FSAK1 F1
C1,FCSK2,FSAK1 F1F6
C1,FCSK2,FSCK1 F2F6

**** Combinacion Diag3: ****
C2,ASM1,FSAK1 F1
C2,FASM1,CK1 F8

C2,FASM1,FSAK1 F1F8
C2,FASM1,FSCK1 F2F8

Entre tanto, el análisis de las funciones adecuadas produce:

```
*****************************************
**** Resultado de la operación Merge: ****
*****************************************
C1,C2,ASK1,CSK2,ASM1,FSAK1 F1
C1,C2,ASK1,CSK2,FASM1,FSAK1 F1F8
C1,C2,ASK1,FCSK2,ASM1,FSAK1 F1F6
C1,C2,ASK1,FCSK2,FASM1,FSAK1 F1F6F8
C1,C2,FASK1,CSK2,ASM1,FSAK1 F1F4
C1,C2,FASK1,CSK2,FASM1,FSAK1 F1F4F8
C1,C2,FASK1,FCSK2,ASM1,FSAK1 F1F4F6
C1,C2,FASK1,FCSK2,FASM1,FSAK1 F1F4F6F8
C1,C2,FASK1,FCSK2,FASM1,FSCK1 F2F4F6F8
|||||||||||||||||||||||||||||||||||||||
**** Resultado de la operación Merge: ****
|||||||||||||||||||||||||||||||||||||||

        **** Combinacion Diag1: ****
        C3,CSK1,CK1 N
        C3,CSK1,FSCK1 F2

        **** Combinacion Diag2: ****
        C3,ASK2,CK1 N
        C3,ASK2,FSCK1 F2

        **** Combinacion Diag3: ****
        C2,ASM1,FSAK1 F1
        C2,FASM1,CK1 F8
        C2,FASM1,FSAK1 F1F8
        C2,FASM1,FSCK1 F2F8

*****************************************
**** Resultado de la operación Merge: ****
```

****
C3,C2,CSK1,ASK2,FASM1,CK1 NF8
C3,C2,CSK1,ASK2,FASM1,FSCK1 F2F8

Resultando en lo que se esperaba:

FCierta encontrada en el proceso Merge por regla de coordinación 3: **F8**

Entonces, esta falla se debe transmitir al Coordinador para que tome las acciones que lleven, al sistema global, a una condición de seguridad.

5.3.2 Caso 2

Para empezar a evaluar el caso 2, se debe conocer los estados que se alcanzan en la condición 1, Figura 5. 6. Siendo:

Diagnosticador 1 → **Id = 2**

C2,ASK1,CK1-t N
C2,ASK1,FSAK1 F1
C2,CSK1-t,FSCK1 F2
C2,FASK1,CK1-t F4
C2,FASK1,CK1 F4
C2,FASK1,FSAK1 F1F4
C2,FASK1,FSCK1 F2F4
C2,FCSK1-t,CK1-t F3
C2,FCSK1-t,FSAK1 F1F3
C2,FCSK1-t,FSCK1 F2F3
C2,CSK1-t,CK1 N
C2,CSK1-t,FSAK1-t F1
C2,FASK1,FSAK1-t F1F4
C2,FCSK1-t,CK1 F3
C2,FCSK1-t,FSAK1-t F1F3
Total pairs = 15
Uncertain: F1 F2 F3 F4
SK1-1 -> 4

Diagnosticador 2 → **Id = 2**

C2,CSK2,CK1-t N

C2,ASK2-t,FSCK1 F2
C2,CSK2,FSAK1 F1
C2,FASK2-t,CK1-t F5
C2,FASK2-t,FSAK1 F1F5
C2,FASK2-t,FSCK1 F2F5
C2,FCSK2,CK1-t F6
C2,FCSK2,CK1 F6
C2,FCSK2,FSAK1 F1F6
C2,FCSK2,FSCK1 F2F6
C2,ASK2-t,CK1 N
C2,ASK2-t,FSAK1-t F1
C2,FASK2-t,CK1 F5
C2,FASK2-t,FSAK1-t F1F5
C2,FCSK2,FSAK1-t F1F6
Total pairs = 15
Uncertain: F1 F2 F5 F6
SK2-0 -> 4

Diagnosticador 3 → **Id = 4**

C3,CSM1,CK1 N
C3,CSM1,FSAK1-t F1
C3,CSM1,FSCK1 F2
C3,FCSM1,CK1 F7
C3,FCSM1,CK1-t F7
C3,FCSM1,FSAK1 F1F7
C3,FCSM1,FSAK1-t F1F7
C3,FCSM1,FSCK1 F2F7
C3,ASM1-t,FSAK1 F1
C3,FASM1-t,CK1 F8
C3,FASM1-t,FSAK1-t F1F8
C3,FASM1-t,FSCK1 F2F8
C3,FASM1-t,FSAK1 F1F8
Total pairs = 13
Uncertain: F1 F2 F7 F8
PK1-0 -> 6
SM1-0 -> 7

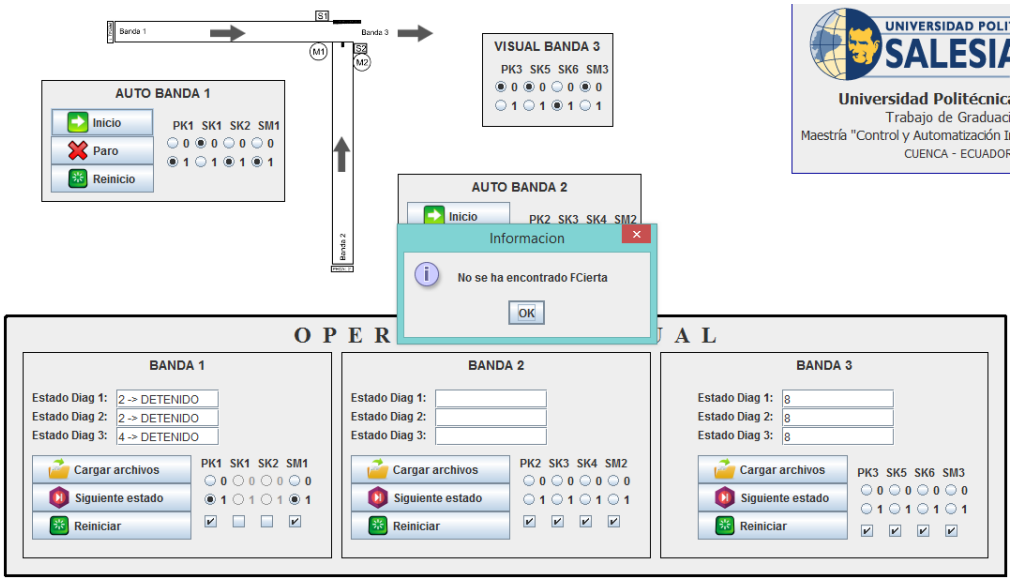


Figura 5. 6 *Resultado de investigación de la primera condición*
Fuente: Autor

Correspondientes a los estados de parada, según los diagramas de la Figura 4. 10 del capítulo 4.

Una vez que se corre el programa, se tiene los estados válidos para el análisis:

**** Combinacion Diag1: ****
C2,ASK1,FSAK1 F1
C2,FASK1,CK1 F4
C2,FASK1,FSAK1 F1F4
C2,FASK1,FSCK1 F2F4

**** Combinacion Diag2: ****
C2,CSK2,FSAK1 F1
C2,FCSK2,CK1 F6
C2,FCSK2,FSAK1 F1F6
C2,FCSK2,FSCK1 F2F6

**** Combinacion Diag3: ****
C3,CSM1,CK1 N
C3,CSM1,FSCK1 F2

C3,FCSM1,CK1 F7
C3,FCSM1,FSAK1 F1F7
C3,FCSM1,FSCK1 F2F7

Entre tanto, el análisis de las funciones adecuadas produce:

****
**** Resultado de la operación Merge: ****
****
C2,C3,ASK1,CSK2,FCSM1,FSAK1 F1F7
C2,C3,ASK1,FCSK2,FCSM1,FSAK1 F1F6F7
C2,C3,FASK1,FCSK2,CSM1,CK1 F4F6N
C2,C3,FASK1,FCSK2,FCSM1,CK1 F4F6F7
C2,C3,FASK1,CSK2,FCSM1,FSAK1 F1F4F7
C2,C3,FASK1,FCSK2,FCSM1,FSAK1 F1F4F6F7
C2,C3,FASK1,FCSK2,CSM1,FSCK1 F2F4F6
C2,C3,FASK1,FCSK2,FCSM1,FSCK1 F2F4F6F7

De lo que se desprende que: No se ha encontrado FCierta

Al momento de introducir la segunda condición, se logra que el algoritmo programado realice los cálculos necesarios, indicados en el capítulo 4 – caso de estudio 2, y genere el resultado demostrado anteriormente.

Alcanzando los estados de llegada, Figura 5. 7, de cada diagnosticador:

Diagnosticador 1 → **Id = 2**

C2,ASK1,CK1-t N
C2,ASK1,FSAK1 F1
C2,CSK1-t,FSCK1 F2
C2,FASK1,CK1-t F4
C2,FASK1,CK1 F4
C2,FASK1,FSAK1 F1F4
C2,FASK1,FSCK1 F2F4
C2,FCSK1-t,CK1-t F3
C2,FCSK1-t,FSAK1 F1F3
C2,FCSK1-t,FSCK1 F2F3
C2,CSK1-t,CK1 N
C2,CSK1-t,FSAK1-t F1
C2,FASK1,FSAK1-t F1F4
C2,FCSK1-t,CK1 F3

C2,FCSK1-t,FSAK1-t F1F3
Total pairs = 15
Uncertain: F1 F2 F3 F4
SK1-1 -> 4

Diagnosticador 2 → **Id = 2**

C2,CSK2,CK1-t N
C2,ASK2-t,FSCK1 F2
C2,CSK2,FSAK1 F1
C2,FASK2-t,CK1-t F5
C2,FASK2-t,FSAK1 F1F5
C2,FASK2-t,FSCK1 F2F5
C2,FCSK2,CK1-t F6
C2,FCSK2,CK1 F6
C2,FCSK2,FSAK1 F1F6
C2,FCSK2,FSCK1 F2F6
C2,ASK2-t,CK1 N
C2,ASK2-t,FSAK1-t F1
C2,FASK2-t,CK1 F5
C2,FASK2-t,FSAK1-t F1F5
C2,FCSK2,FSAK1-t F1F6
Total pairs = 15
Uncertain: F1 F2 F5 F6
SK2-0 -> 4

Diagnosticador 3 → **Id = 8**

C1,ASM1,AK1 N
C1,FCSM1-t,AK1 F7
C1,FASM1,AK1 F8
C1,ASM1,FSAK1 F1
C1,ASM1,FSCK1-t F2
C1,FCSM1-t,FSAK1 F1F7
C1,FASM1,FSAK1 F1F8
C1,FCSM1-t,FSCK1-t F2F7
C1,FASM1,FSCK1-t F2F8
C1,CSM1-t,FSCK1 F2
C1,FASM1,AK1-t F8
C1,FASM1,FSCK1 F2F8
C1,FCSM1-t,FSCK1 F2F7
Total pairs = 13
Uncertain: F1 F2 F7 F8
PK1-1 -> 2
SM1-1 -> 3

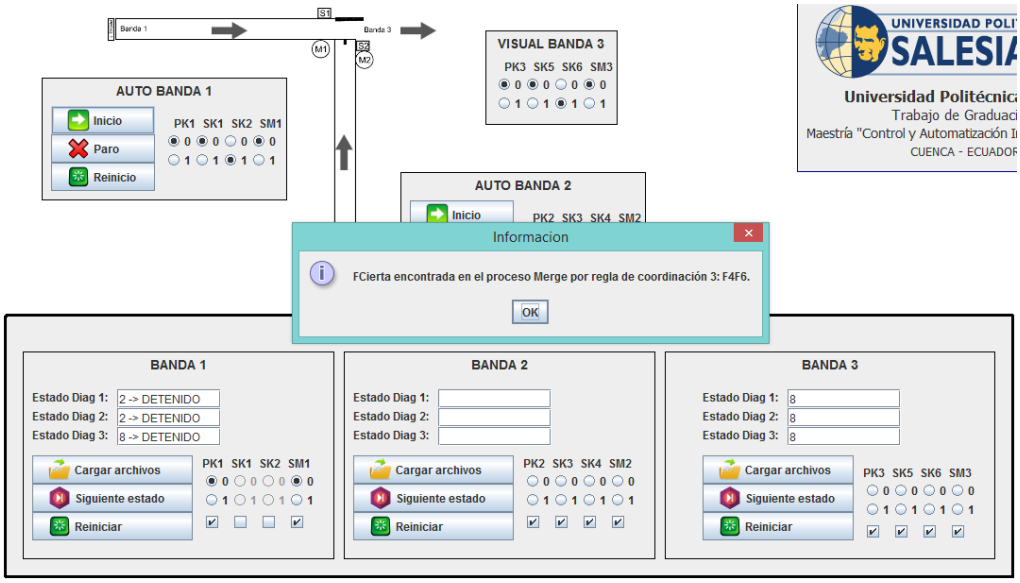


Figura 5. 7 *Resultado final, con la segunda condición*
Fuente: Autor

El programa arroja los resultados que validan la falla presentada:

**** Combinacion Diag1: ****
C2,ASK1,FSAK1 F1
C2,FASK1,CK1 F4
C2,FASK1,FSAK1 F1F4
C2,FASK1,FSCK1 F2F4

**** Combinacion Diag2: ****
C2,CSK2,FSAK1 F1
C2,FCSK2,CK1 F6
C2,FCSK2,FSAK1 F1F6
C2,FCSK2,FSCK1 F2F6

**** Combinacion Diag3: ****
C1,ASM1,AK1 N
C1,FASM1,AK1 F8
C1,ASM1,FSAK1 F1
C1,FASM1,FSAK1 F1F8

C1,FASM1,FSCK1 F2F8

Entre tanto, el análisis de las funciones adecuadas produce:

```
*****************************************
**** Resultado de la operación Merge: ****
*****************************************
C2,C1,ASK1,CSK2,ASM1,FSAK1 F1
C2,C1,ASK1,CSK2,FASM1,FSAK1 F1F8
C2,C1,ASK1,FCSK2,ASM1,FSAK1 F1F6
C2,C1,ASK1,FCSK2,FASM1,FSAK1 F1F6F8
C2,C1,FASK1,CSK2,ASM1,FSAK1 F1F4
C2,C1,FASK1,CSK2,FASM1,FSAK1 F1F4F8
C2,C1,FASK1,FCSK2,ASM1,FSAK1 F1F4F6
C2,C1,FASK1,FCSK2,FASM1,FSAK1 F1F4F6F8
C2,C1,FASK1,FCSK2,FASM1,FSCK1 F2F4F6F8
|||||||||||||||||||||||||||||||||||||||
**** Resultado de la operación Merge: ****
|||||||||||||||||||||||||||||||||||||||

        **** Combinacion Diag1: ****
        C2,ASK1,FSAK1 F1
        C2,FASK1,CK1 F4
        C2,FASK1,FSAK1 F1F4
        C2,FASK1,FSCK1 F2F4

        **** Combinacion Diag2: ****
        C2,CSK2,FSAK1 F1
        C2,FCSK2,CK1 F6
        C2,FCSK2,FSAK1 F1F6
        C2,FCSK2,FSCK1 F2F6

        **** Combinacion Diag3: ****
        C3,CSM1,CK1 N
        C3,CSM1,FSCK1 F2
*****************************************
**** Resultado de la operación Merge: ****
*****************************************
C2,C3,FASK1,FCSK2,CSM1,CK1 F4F6N
C2,C3,FASK1,FCSK2,CSM1,FSCK1 F2F4F6
```

Resultando en lo que se esperaba:

FCierta encontrada en el proceso Merge por regla de coordinación 3: **F4F6**

Entonces, esta falla se debe transmitir al Coordinador para que tome las acciones que lleven, al sistema global, a una condición de seguridad.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

Al final, al mirar en retrospectiva todo el trabajo realizado, y considerando los argumentos iniciales, se logra concluir que:

- Se ha alcanzado el objetivo general, logrando construir un diagnosticador descentralizado para la detección de fallas basado en eventos discretos, para ser usado en un automatismo de bandas transportadoras. Para lo cual, un diagnosticador monolítico se pudo separar en diagnosticadores individuales locales, los mismos que se enlazaban por un elemento común (actuador).

- Se desarrollaron los autómatas requeridos por medio del software Desuma V3.0, de la Universidad de Michigan. A pesar de ciertas limitaciones de edición, es una herramienta muy adecuada para el análisis previo de un autómata. Llegándose a intuir la complejidad de un autómata monolítico, el cual tiene un crecimiento dramático de estados al darse una composición paralela de los varios sensores que intervienen con los actuadores respectivos.

- Con respecto al modelado, se debe considerar todas las opciones de operación del automatismo. Y para ésto, se debe de conocer cómo realmente funciona. Esto se logra con el verdadero conocimiento de la maquinaria, de los elementos que se encuentren fungiendo de sensores y de actuadores; y, de cómo se enlazan en su accionar para controlar al automatismo en sí.

  Entonces, con todas estas ideas, se plantea el autómata que requiere determinado actuador, así mismo, se plantea un autómata de operación para cada sensor. Luego, de acuerdo al autómata de control monolítico, se diseña los autómatas de control para cada sensor.

  La combinación paralela de estos autómatas permite tener un autómata diagnosticador, que verifica las posibles fallas que se producen en el equipo respectivo. Pero hay que tener sumo cuidado en las condicionantes que se den al momento de diseñar el autómata del actuador y de los sensores. Dado que sólo se puede llevar a cabo lo que se modela, y para cualquier otro cambio que

se desee introducir en la manera de operación de estos, se debe retornar a los autómatas de operación iniciales y modelarlo.

- Dentro de los ambientes de simulación, se debe escoger una plataforma de programación que permita integrar fácilmente cada autómata. Pero esto va a estar sujeto a la experticia del programador. Basado en esto, en este trabajo se tomó al software Java orientado a objetos, y se empleó lo que se indica en la Buena Práctica de Programación, es decir, trabajarlo por módulos (subrutinas).

  En el mercado industrial se cuenta con empresas que, de alguna manera, ya refieren con procedimientos estandarizados para que sus equipos operen con este sistema de autómatas.

- Por último, como puntos clave del análisis propuesto en este trabajo, se tiene:

  - Todas las fallas planteadas se consideran "constantes", lo que indica que al darse, la única forma de retirarlas es actuando sobre ella en el elemento físico en donde se genera.
  - Dado por el tipo de modelado empleado, la falla sólo puede darse cuando el elemento físico está estable en su operación, y no en el paso de un estado a otro.
  - Dado que se trabaja con elementos de control físico (reales), se debe considerar que estos pueden presentar malfuncionamiento durante su vida de trabajo.
  - Todo evento de falla se asume que ocurre luego de un tiempo de espera permitido (time – out), que una vez superado, se realiza los análisis respectivos a partir de los eventos observados de los sensores, para aislar la falla y avisar a un sistema central para que tome las correcciones del caso.

6.2 Recomendaciones

- Efectuar análisis (implementación de este modo de detectar fallos en un sistema) más extensos en sistemas de automatización más complejos; para determinar su validez en un rango amplio de automatismos industriales. Evaluando lo que sucede cuando una falla se produce en el momento de la transición de un equipo.

- Estandarizar los métodos de detección de fallas para que sean más amigables, referente a la programación en softwares que efectúen control industrial. Esto permitirá un enfoque descentralizado siempre, lo que conllevaría a una confiabilidad más certera sobre la programación y manipulación de los equipos industriales.

ANEXO

Anexo 1: Codificación de estados y eventos

- La letra “A” al inicio de cada etiqueta simboliza que el dispositivo está “abierto”, lo que se representa para fines prácticos como “0L”.
- La letra “C” al inicio de cada etiqueta simboliza que el dispositivo está “cerrado”, lo que se representa para fines prácticos como “1L”.
- La letra “F” al inicio de cada etiqueta simboliza que el dispositivo está en “falla”.

CÓDIGO	**DESCRIPCIÓN DE LOS ESTADOS**
AK1	actuador 1 abierto: K1
AK1-t	transitorio de cerrado a abierto de actuador 1
CK1	actuador 1 cerrado: K1
CK1-t	transitorio de abierto a cerrado de actuador 1
FSAK1	estado de falla: atasco de actuador K1 en abierto
FSAK1-t	estado de falla transitorio: atasco de actuador K1 en abierto
FSCK1	estado de falla: atasco de actuador K1 en cerrado
FSCK1-t	estado de falla transitorio: atasco de actuador K1 en cerrado
AK2	actuador 2 abierto: K2
AK2-t	transitorio de cerrado a abierto de actuador 2
CK2	actuador 2 cerrado: K2
CK2-t	transitorio de abierto a cerrado de actuador 2
FSAK2	estado de falla: atasco de actuador K2 en abierto
FSAK2-t	estado de falla transitorio: atasco de actuador K2 en abierto
FSCK2	estado de falla: atasco de actuador K2 en cerrado
FSCK2-t	estado de falla transitorio: atasco de actuador K2 en cerrado
AK3	actuador 3 abierto: K3
AK3-t	transitorio de cerrado a abierto de actuador 3
CK3	actuador 3 cerrado: K3

CK3-t	transitorio de abierto a cerrado de actuador 3
FSAK3	estado de falla: atasco de actuador K3 en abierto
FSAK3-t	estado de falla transitorio: atasco de actuador K3 en abierto
FSCK3	estado de falla: atasco de actuador K3 en cerrado
FSCK3-t	estado de falla transitorio: atasco de actuador K3 en cerrado
ASK1	sensor 1 abierto: SK1
ASK1-t	transitorio de cerrado a abierto de sensor 1
CSK1	sensor 1 cerrado: SK1
CSK1-t	transitorio de abierto a cerrado de sensor 1
FASK1	estado de falla: atasco de sensor 1 en abierto
FASK1-t	estado de falla transitorio: atasco de sensor 1 en abierto
FCSK1	estado de falla: atasco de sensor 1 en cerrado
FCSK1-t	estado de falla transitorio: atasco de sensor 1 en cerrado
ASK2	sensor 2 abierto: SK2
ASK2-t	transitorio de cerrado a abierto de sensor 2
CSK2	sensor 2 cerrado: SK2
CSK2-t	transitorio de abierto a cerrado de sensor 2
FASK2	estado de falla: atasco de sensor 2 en abierto
FASK2-t	estado de falla transitorio: atasco de sensor 2 en abierto
FCSK2	estado de falla: atasco de sensor 2 en cerrado
FCSK2-t	estado de falla transitorio: atasco de sensor 2 en cerrado
ASM1	sensor 3 abierto: SM1
ASM1-t	transitorio de cerrado a abierto de sensor 3
CSM1	sensor 3 cerrado: SM1
CSM1-t	transitorio de abierto a cerrado de sensor 3
FASM1	estado de falla: atasco de sensor 3 en abierto
FASM1-t	estado de falla transitorio: atasco de sensor 3 en abierto
FCSM1	estado de falla: atasco de sensor 3 en cerrado
FCSM1-t	estado de falla transitorio: atasco de sensor 3 en cerrado

ASK3	sensor 4 abierto: SK3
ASK3-t	transitorio de cerrado a abierto de sensor 4
CSK3	sensor 4 cerrado: SK3
CSK3-t	transitorio de abierto a cerrado de sensor 4
FASK3	estado de falla: atasco de sensor 4 en abierto
FASK3-t	estado de falla transitorio: atasco de sensor 4 en abierto
FCSK3	estado de falla: atasco de sensor 4 en cerrado
FCSK3-t	estado de falla transitorio: atasco de sensor 4 en cerrado
ASK4	sensor 5 abierto: SK4
ASK4-t	transitorio de cerrado a abierto de sensor 5
CSK4	sensor 5 cerrado: SK4
CSK4-t	transitorio de abierto a cerrado de sensor 5
FASK4	estado de falla: atasco de sensor 5 en abierto
FASK4-t	estado de falla transitorio: atasco de sensor 5 en abierto
FCSK4	estado de falla: atasco de sensor 5 en cerrado
FCSK4-t	estado de falla transitorio: atasco de sensor 5 en cerrado
ASM2	sensor 6 abierto: SM2
ASM2-t	transitorio de cerrado a abierto de sensor 6
CSM2	sensor 6 cerrado: SM2
CSM2-t	transitorio de abierto a cerrado de sensor 6
FASM2	estado de falla: atasco de sensor 6 en abierto
FASM2-t	estado de falla transitorio: atasco de sensor 6 en abierto
FCSM2	estado de falla: atasco de sensor 6 en cerrado
FCSM2-t	estado de falla transitorio: atasco de sensor 6 en cerrado
ASK5	sensor 7 abierto: SK5
ASK5-t	transitorio de cerrado a abierto de sensor 7
CSK5	sensor 7 cerrado: SK5
CSK5-t	transitorio de abierto a cerrado de sensor 7
FASK5	estado de falla: atasco de sensor 7 en abierto

FASK5-t	estado de falla transitorio: atasco de sensor 7 en abierto
FCSK5	estado de falla: atasco de sensor 7 en cerrado
FCSK5-t	estado de falla transitorio: atasco de sensor 7 en cerrado
ASK6	sensor 8 abierto: SK6
ASK6-t	transitorio de cerrado a abierto de sensor 8
CSK6	sensor 8 cerrado: SK6
CSK6-t	transitorio de abierto a cerrado de sensor 8
FASK6	estado de falla: atasco de sensor 8 en abierto
FASK6-t	estado de falla transitorio: atasco de sensor 8 en abierto
FCSK6	estado de falla: atasco de sensor 8 en cerrado
FCSK6-t	estado de falla transitorio: atasco de sensor 8 en cerrado
ASM3	sensor 9 abierto: SM3
ASM3-t	transitorio de cerrado a abierto de sensor 9
CSM3	sensor 9 cerrado: SM3
CSM3-t	transitorio de abierto a cerrado de sensor 9
FASM3	estado de falla: atasco de sensor 9 en abierto
FASM3-t	estado de falla transitorio: atasco de sensor 9 en abierto
FCSM3	estado de falla: atasco de sensor 9 en cerrado
FCSM3-t	estado de falla transitorio: atasco de sensor 9 en cerrado

Tabla A. 1 *Codificación de los estados de cada actuador y sensor, y de los eventos en cada autómata*
Fuente: Autor

Anexo 2: Codificación de fallas

FALLA	CÓDIGO	DESCRIPCIÓN DE LA FALLA	POSICIÓN EN BANDA
F30	so1	atasco en abierto de actuador 1	1
F31	sc1	atasco en cerrado de actuador 1	1
F40	so2	atasco en abierto de actuador 2	2
F41	sc2	atasco en cerrado de actuador 2	2
F50	so3	atasco en abierto de actuador 3	3
F51	sc3	atasco en cerrado de actuador 3	3
F1	g1	atasco en cerrado sensor SK1	1
F2	g2	atasco en abierto sensor SK1	1
F3	g3	atasco en abierto sensor SK2	1
F4	g4	atasco en cerrado sensor SK2	1
F5	g5	atasco en cerrado sensor SM1	1
F6	g6	atasco en abierto sensor SM1	1
F10	g7	atasco en abierto sensor SK3	2
F11	g8	atasco en cerrado sensor SK3	2
F12	g9	atasco en abierto sensor SK4	2
F13	g10	atasco en cerrado sensor SK4	2
F14	g11	atasco en cerrado sensor SM2	2
F15	g12	atasco en abierto sensor SM2	2
F20	g13	atasco en abierto sensor SK5	3
F21	g14	atasco en cerrado sensor SK5	3
F22	g15	atasco en abierto sensor SK6	3
F23	g16	atasco en cerrado sensor SK6	3
F24	g17	atasco en abierto sensor SM3	3
F25	g18	atasco en cerrado sensor SM3	3

Tabla A. 2 *Codificación de fallas*
Fuente: Autor

Anexo 3: Codificación de los estados de cada diagnosticador

Estados para diagnosticador 1:

```
Total Diagnoser States = 9

Id = 1
C1,ASK1,AK1 N
C1,FCSK1-t,AK1 F3
C1,FASK1,AK1 F4
C1,ASK1,FSAK1 F1
C1,ASK1,FSCK1-t F2
C1,FCSK1-t,FSAK1 F1F3
C1,FASK1,FSAK1 F1F4
C1,FCSK1-t,FSCK1-t F2F3
C1,FASK1,FSCK1-t F2F4
C1,CSK1-t,FSCK1 F2
C1,FASK1,FSCK1 F2F4
C1,FCSK1-t,FSCK1 F2F3
Total pairs = 12
Uncertain: F1 F2 F3 F4
PK1-1 -> 2
SK1-1 -> 3

Id = 2
C2,ASK1,CK1-t N
C2,ASK1,FSAK1 F1
C2,CSK1-t,FSCK1 F2
C2,FASK1,CK1-t F4
C2,FASK1,CK1 F4
C2,FASK1,FSAK1 F1F4
C2,FASK1,FSCK1 F2F4
C2,FCSK1-t,CK1-t F3
C2,FCSK1-t,FSAK1 F1F3
C2,FCSK1-t,FSCK1 F2F3
C2,CSK1-t,CK1 N
C2,CSK1-t,FSAK1-t F1
C2,FASK1,FSAK1-t F1F4
C2,FCSK1-t,CK1 F3
C2,FCSK1-t,FSAK1-t F1F3
Total pairs = 15
Uncertain: F1 F2 F3 F4
SK1-1 -> 4
Id = 3
```

```
E1,CSK1,FSCK1 F2
E1,FCSK1,AK1 F3
E1,FCSK1,FSAK1 F1F3
E1,FCSK1,FSCK1 F2F3
E1,FCSK1,FSCK1-t F2F3
E1,FASK1-t,FSCK1 F2F4
Total pairs = 6
Uncertain: F1 F2 F3 F4
SK1-0 -> 5

Id = 4
C3,CSK1,CK1 N
C3,CSK1,FSAK1-t F1
C3,CSK1,FSCK1 F2
C3,FCSK1,CK1 F3
C3,FCSK1,CK1-t F3
C3,FCSK1,FSAK1 F1F3
C3,FCSK1,FSAK1-t F1F3
C3,FCSK1,FSCK1 F2F3
C3,ASK1-t,FSAK1 F1
C3,FASK1-t,CK1 F4
C3,FASK1-t,FSAK1-t F1F4
C3,FASK1-t,FSCK1 F2F4
C3,FASK1-t,FSAK1 F1F4
Total pairs = 13
Uncertain: F1 F2 F3 F4
PK1-0 -> 6
SK1-0 -> 7

Id = 5
E1,FASK1,FSCK1 F2F4
Total pairs = 1
Certain: F2 F4

Id = 6
C4,ASK1-t,FSAK1 F1
C4,CSK1,AK1-t N
C4,CSK1,FSCK1 F2
C4,FASK1-t,AK1-t F4
C4,FASK1-t,FSAK1 F1F4
C4,FASK1-t,FSCK1 F2F4
C4,FCSK1,AK1-t F3
C4,FCSK1,AK1 F3
C4,FCSK1,FSAK1 F1F3
```

```
C4,FCSK1,FSCK1 F2F3
C4,ASK1-t,AK1 N
C4,ASK1-t,FSCK1-t F2
C4,FASK1-t,AK1 F4
C4,FASK1-t,FSCK1-t F2F4
C4,FCSK1,FSCK1-t F2F3
Total pairs = 15
Uncertain: F1 F2 F3 F4
SK1-0 -> 8

Id = 7
E1,ASK1,FSAK1 F1
E1,FASK1,CK1 F4
E1,FASK1,FSAK1 F1F4
E1,FASK1,FSAK1-t F1F4
E1,FASK1,FSCK1 F2F4
E1,FCSK1-t,FSAK1 F1F3
Total pairs = 6
Uncertain: F1 F2 F3 F4
SK1-1 -> 9

Id = 8
C1,ASK1,AK1 N
C1,FCSK1-t,AK1 F3
C1,FASK1,AK1 F4
C1,ASK1,FSAK1 F1
C1,ASK1,FSCK1-t F2
C1,FCSK1-t,FSAK1 F1F3
C1,FASK1,FSAK1 F1F4
C1,FCSK1-t,FSCK1-t F2F3
C1,FASK1,FSCK1-t F2F4
C1,CSK1-t,FSCK1 F2
C1,FASK1,AK1-t F4
C1,FASK1,FSCK1 F2F4
C1,FCSK1-t,FSCK1 F2F3
Total pairs = 13
Uncertain: F1 F2 F3 F4
PK1-1 -> 2
SK1-1 -> 3

Id = 9
E1,FCSK1,FSAK1 F1F3
Total pairs = 1
Certain: F1 F3
```

BIBLIOGRAFÍA

Banks, J. y., & et_al. (2004). *Discrete - Event System Simulation.* New Jersey: Prentice Hall.

Basilio, J., Carvalho, L., & Moreira, M. (2010). Diagnose de falhas em sistemas a eventos discretos modelados por autómatos finitos. *Controle & Automacao*, Vol. 21 ; pag. 510 - 533.

Blanke, M. (septiembre de 1996). *Consistent design of dependable control systems.* Obtenido de ScienceDirect: http://www.sciencedirect.com/science/article/pii/0967066196001384

Bloch, J. (2008). *Effective Java.* Santa Clara: Addison - Wesley.

Bock, H. (2012). *The Definitive Guide to NetBeans Platform 7.* Apress.

Cerrada, M. (2014). Sistemas de eventos discretos, introducción. Cuenca, Azuay, Ecuador.

Cerrada, M., Ferrarini, L., & Dedé, A. (2010). *Modular fault diagnosis using temporized analysis for a class of discrete event systems.* Venezuela: Universidad de los Andes.

Contant, O., Lafortune, S., & Teneketzis, D. (2006). *Diagnosability of Discrete Event Systems with Modular Structure.* Obtenido de University's Michigan: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.1974&rep=rep1&type=pdf

Debouk, R. (25 de noviembre de 2016). *https://www.researchgate.net/.* Obtenido de https://www.researchgate.net/: https://www.researchgate.net/publication/4042986_Diagnosis_of_discrete_event_systems_A_modular_approach

Eckel, B. (2006). *Thinking in Java.* New Jersey: Prentice Hall.

Epp, S. (2011). *Matemáticas Discretas con Aplicaciones.* México: Cengage Learning.

Fargo Control, I. (16 de 02 de 2017). *http://www.fargocontrols.com/.* Obtenido de http://www.fargocontrols.com/pdf/controls/Fargo-RC_N.pdf

Ferrarini, L., Allevi, M., & A., D. (02 de september de 2011). *A real-time algorithm for fault identification in machining centres.* doi:10.3182/20110828-6-IT-1002.02021

Ferrarini, L., Brusa, R., & Veber, C. (2008). *A pragmatic approach to fault diagnosis in hydraulic circuits for automated machining: A case study.* Arlington: IEEE International Conference on Automation Science and Engineering.

Gascar, E., & Zineb, S.-A. (2011). Automatic Construction of Diagnoser for Complex Discrete Event Systems. Saarbrucken, Germany. doi:hal-00676764

González, O., & Cerrada, M. (2014). *Diagnóstico de sistemas de eventos discretos controlados: Un enfoque basad en crónicas y análisis modular usando modelos de autómatas.* doi:10.1016/j.riai.2010.02.003

Hopcroft, J., & Ullman, J. (1993). *Introducción a la Teoría de Autómatas, Lenguajes y Computación.* México: CECSA.

Isasi, P., Martínez, P., & Borrajo, D. (1997). *Lenguajes, Gramáticas y Autómatas: un enfoque práctico.* Madrid: Addison - Wesley.

Kelley, D. (1995). *Teoría de autómatas y lenguajes formales.* Madrid: Pearson Educación.

Lafortune, S., & Cassandras, C. (2008). *Introduction to Discret Event Systems.* New York: Springer.

Law, A., & Kelton, D. (1991). *Simulation Modeling and Analysis.* Singapore: McGraw-Hill.

Llobet , R. e. (2009). *Introducción a la Programación Orientada a Objetos con Java.* Valencia: Universidad Politécnica de Valencia.

Maquinariaspesadas.org. (26 de Enero de 2016). *https://www.maquinariaspesadas.org.* Obtenido de https://www.maquinariaspesadas.org: https://www.maquinariaspesadas.org/blog/2490-manual-cintas-transportadoras-tipos-estructura-operaciones-aplicaciones

Monsalve M., O. E. (2011). *Diagnóstico de fallas para una clase de sistemas a eventos discretos usando un enfoque modular.* Mérida, Venezuela: Universidad de los Andes.

Morant, F., Quiles, E., Blasco, R., & Correcher, A. (2001). *Diágnostico de fallos de una columna de detilación.* Universidad Politécnica de Valencia. Obtenido de http://intranet.ceautomatica.es/old/actividades/jornadas/XXII/documentos/C_01_CI.pdf

Ocheana, L., Popescu, D., & Ferrarini, L. (2013). Group-based method for fault identification using the diagnoser approach. *CEAI, 15*(4), 66-76. Obtenido de http://www.ceai.srait.ro/index.php?journal=ceai&page=article&op=view&path[]=1687&path[]=1270

Ortíz, J. (2015). *Diseño de un sistema de diagnóstico de fallas basado en eventos discretos para un sistema de nivel de líquido.* Cuenca - Azuay - Ecuador: Universidad Politécnica Salesiana.

Ricker, S., & Fabre, E. (2000). *On the construction of modular observers and diagnosers for discrete-event systems.* The Pennsylvania State University. doi:10.1109/CDC.2000.914130

Rivera, M., & Manuel, H. (2007). *Diagnóstico de fallas en Sistemasa Eventos Discretos: Una Propuesta de aplicación en Procesos de Separación de Óleo - Gas.* Río de Janeiro: Universidad Federal de Río de Janeiro. Recuperado el 15 de 10 de 2016

Salinero, G. M. (2013). *Diseño de una banda transportadora mediante guide de matlab.* Madrid: Universidad Carlos III de Madrid.

Sampath, M. (1995). *A Discrete Event Systems Approach to Failure Diagnosis.* Michigan: University of Michigan.

Sayed-Mouchaweh, M. (2014). *Discrete Event Systems: Diagnosis and Diagnosability.* New York: Springer.

Zhou, C., & et_al. (2008). *Decentralized Modular Diagnosis of Concurrent Discrete Event Systems.* Iowa. doi:10.1109/WODES.200804605977