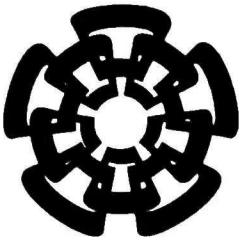
BC-662
Don-2012

xx(182693.1)



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Implementación Formal de Supervisores Jerárquico-Modulares en Aplicaciones de Manufactura

Tesis que presenta:
Jorge Alfonso Bucio Cisneros

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Arturo del Sagrado Corazón Sánchez Carmona

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Agosto de 2011.

ID - 181196-1001

Implementación Formal de Supervisores Jerárquico-Modulares en Aplicaciones de Manufactura

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Jorge Alfonso Bucio Cisneros
Ingeniero en Electrónica y Comunicaciones
Instituto Tecnológico y de Estudios Superiores de Monterrey
2004-2008

Becario de Conacyt, expediente no. 234880

Director de Tesis
Dr. Arturo del Sagrado Corazón Sánchez Carmona

CINVESTAV del IPN Unidad Guadalajara, Agosto de 2011.

Tesis de Maestría en Ciencias en Ingeniería Eléctrica

Presentada por:

Jorge Alfonso Bucio Cisneros

Para obtener el grado de:

Maestro en Ciencias

Con especialidad en:

Ingeniería Eléctrica

Dr. Eduardo Aranda Bricaire
Sinodal

Dr. Andrés Méndez Vázquez
Sinodal

Dr. Arturo del Sagrado Corazón Sánchez Carmona

Director de Tesis

11 de Julio de 2011

Resumen

En este trabajo de tesis se realiza un estudio de dos técnicas de control jerárquico-modular: la *vocalización* desarrollada por el Profesor W.M. Wonham y la técnica de *control por eventos imperativos* expuesta en el trabajo de tesis de F. Jaimes; resaltando las ventajas e inconvenientes de cada una en aplicaciones de manufactura así como sus contribuciones frente a otras metodologías. Se utilizan autómatas como formalismo de modelado y se adopta la Teoría de Control Supervisor.

Se propone un esquema de traducción para llevar a la implementación los supervisores sintetizados con cada técnica. Aunque existen otras propuestas de implementación de supervisores y herramientas de generación automática de código, ninguna de éstas corresponden a implementaciones jerárquicas ni los supervisores son empleados como dispositivos de control.

Así mismo se presenta una propuesta para la implementación de la capa de comunicación entre los supervisores y el sistema a controlar, estableciendo una traducción entre las Máquinas de Estados Finitos que modelan a los componentes elementales del sistema y diagrama de escalera, para cualquier sistema en general. Se presenta el diseño e implementación de las diferentes estructuras de control jerárquicas y modulares aplicadas en un prototipo de sistema automatizado de manufactura (SAM) hecho en Lego®. Se hizo la integración de trabajos anteriores relacionados con el SAM, se diseñó otra versión de interfaz electrónica, destinada al acondicionamiento de señales y a la comunicación entre el prototipo y el sistema de control (PC y PLC Siemens Simatic S7-300). El modelado de las estructuras del prototipo se basó en el modelo híbrido construido con los estándares industriales ISA-S88 e ISA-S95.

El trabajo culmina con un análisis comparativo entre las dos técnicas, analizando las restricciones y condiciones en el modelado que cada técnica exige, la flexibilidad de la estructura resultante ante los cambios en el sistema o en el plan de producción y su complejidad y uso de recursos en la implementación. Resultando ser la técnica de *Control por eventos imperativos* la que reúne mayores ventajas en aplicaciones de manufactura; como por ejemplo, la implementación de los supervisores empleando esta técnica emplea menos recursos en el PLC.

Abstract

This thesis presents a study of two techniques of hierarchical control for discrete-event systems: one of them based on *vocalized events*, developed by W.M. Wonham and the other based on *imperative events*, proposed by F. Jaime in his M. Sc. thesis. This work establishes advantages and disadvantages of applying these techniques to automated manufacturing systems as well as their contributions compared with other methodologies.
A translation scheme is proposed in order to implement the synthesized supervisors with both techniques. Although there are other implementation approaches and automatic code generation tools, none of them corresponds to hierarchical implementations and neither the supervisors are used as control devices.
A proposal for implementing the I/O layer between the supervisors and the controlled system is also presented. A translation scheme is put forward from Finite State Machines modeling the elementary components of the system to their representation as ladder diagrams.

The design and implementation of the different hierarchical control structures were applied to an automated manufacturing system (AMS) prototype built in Lego. Capitalizing on previous works related with the AMS Lego, a new version of the electronic interface was designed, intended for signal conditioning and communicating the prototype with the control system (PC and PLC Siemens Simatic S7-300). The prototype structures were based on the hybrid model designed with the industrial standards ISA-S88 and ISA-S95.

The document concludes with a comparative analysis between both techniques, studying the constraints and modeling conditions, the resultant structure flexibility to changes in system topology or production plans and the use of resources of the PLC in the implementation. The technique based on *imperative events* exhibited a larger number of advantages for manufacturing applications; for example, the supervisor implementation using this technique employs less PLC resources.

Agradecimientos

Son muchas las personas que han participado en este trabajo aunque sea de manera indirecta y a quienes quiero expresar mi gratitud por el apoyo y la confianza que me han prestado de forma desinteresada.

En primer lugar quiero agradecer al Dr. Arturo Sánchez Carmona por el tiempo que me ha brindado, por sus sugerencias e ideas y por la dirección de este trabajo.

Gracias al Centro de Investigación y Estudios Superiores del Politécnico Nacional y al departamento de Ingeniería de Control por su acogida y apoyo durante este período en mi desarrollo formativo, a todos mis compañeros y amigos de esta institución, en especial a la generación 2009 y al CONACYT por la beca número 234880.

Me complace agradecer también a mis compañeros de la generación 2010 por su aportación con las pruebas en el prototipo de sistema automatizado de manufactura brindando retroalimentación y sugerencias.

Un sincero agradecimiento a mis amigos José de Jesús Camacho y Guillermo Palomino por prestarme las instalaciones del Centro de Diseño Electrónico del ITESM y por su soporte en el proceso de diseño del PCB.

Debo un especial reconocimiento a mis padres por haberme dado la oportunidad de tener una educación superior.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Un Sistema de Eventos Discretos (SED) es un sistema dinámico, con un conjunto de estados numerable, donde el estado cambia repentinamente por la ocurrencia de eventos generalmente asíncronos en el tiempo. Un evento puede ser identificado con una acción específica como por ejemplo, el presionar un botón o una ocurrencia dictada por un programa.
Muchos sistemas son SED's; como por ejemplo, los sistemas computacionales (el uso de un recurso en una computadora), los sistemas de comunicación (uso de canales de comunicación responsable de la transmisión de mensajes entre procesadores o dispositivos periféricos), los sistemas de bases de datos (el acceso a una base de datos), así como los sistemas de manufactura (donde varios tipos de máquinas o robots son utilizados en los procesos de manufactura).

Numerosos investigadores han desarrollado estrategias para el control de SED's. Las teorías desarrolladas y reportadas por ellos contienen entre otras partes, definiciones de la propiedad de controlabilidad, condiciones para determinar si un sistema puede ser controlado y el diseño de agentes que restrinjan el comportamiento del sistema al comportamiento requerido.

En 1986 P.J. Ramadge yW. M. Wonham en [1] proponen la Teoría de Control Supervisor (TCS). El objetivo de la TCS es restringir el comportamiento del sistema en un conjunto de secuencias de eventos requeridos (especificación). En esta teoría, tanto el comportamiento del sistema como el comportamiento requerido son descritos por lenguajes formales o por autómatas.

Otro resultado importante en la TCS es que la tarea de control es formulada como la síntesis de un supervisor [2], que deshabilita los eventos controlables del sistema cuando sea necesario, para confinar el comportamiento del sistema al de la especificación [3].

Un problema en la TCS es la presencia de bloqueo en el sistema bajo control. Los bloqueos ocurren cuando la secuencia que está realizando el SED controlado no puede concatenarse con alguna otra secuencia, de manera que llegue a un estado que represente que la tarea ha sido completada. Una de las primeras nociones exploradas por los in-

vestigadores en control de SED's enfocados en soluciones no bloqueantes basado en la TCS, fue el estudio del sublenguaje supremo controlable propuesto en [4]. Esta alternativa propone obtener una nueva especificación que contenga todas las secuencias requeridas originalmente, removiendo aquellas secuencias que no producen tareas completas en el sistema o que no son controlables. Otra solución es propuesta en [5], donde el papel central es caracterizar una solución que involucre un compromiso entre satisfacción y bloqueo.

Por otra parte, debido a que la mayoría de los SED's tienen un gran número de componentes, y que éstos al ser modelados como autómatas finitos generan un espacio de estados muy grande, determinar si una especificación es controlable era una tarea compleja. Tratando de evitar este inconveniente, en [6] se introdujo el *control supervisor modular*. En este enfoque, el comportamiento requerido (especificación global) es expresado como la conjunción de un conjunto de predicados (o módulos) que el sistema debe satisfacer y para cada módulo es calculado un supervisor. Se considera que cada supervisor tiene acceso a toda la información del sistema. Además es necesario calcular un supervisor central (es decir un supervisor global) como la conjunción de los supervisores para cada módulo. Este supervisor central, provoca que el sistema realice las secuencias comunes a todas las especificaciones.

Otra aproximación similar a supervisión modular basada en TCS es el *supervisor descentralizado*, propuesto por Lin y Wonham en [7]. Esa aproximación se basa en el concepto de "divide y vencerás", en el cual el sistema se particiona en n subsistemas, calculando un supervisor para cada subsistema. Posteriormente, la conexión entre los supervisores locales deberá ser realizada por un *coordinador*. En otras palabras, el comportamiento global puede reducirse a la satisfacción simultánea de las especificaciones locales. La idea principal de supervisión descentralizada, es que los supervisores locales simultáneamente restrinjan el comportamiento del sistema al comportamiento de la especificación (en donde cada supervisor tiene acceso sólo a la información local) La descentralización impone un control en dos niveles: en el primer nivel cada supervisor es usado para controlar cada sección del sistema; el segundo usa un *coordinador* para acoplar el comportamiento de cada supervisor para evitar *conflictos* [7], [8], [9], [10], [11].

Años más tarde se articularon todos los conocimientos dando lugar al esquema de *descomposición arquitectural* (modularización horizontal y vertical) para el diseño de sistemas de control complejos. Se proponen arquitecturas *jerárquicas, modulares y descentralizadas* [12], [13], [14]. En la modularización vertical, Zhong y Wonham consideran dos niveles de jerarquía, en donde el sistema superior es un modelo agregado del proceso del nivel bajo, el cual es manejado mediante un canal de información. Se introduce el concepto de *consistencia jerárquica* para asegurar que la tarea comandada por el controlador del nivel superior pueda ser realizada en el nivel inferior, lo que significa que la propiedad de no bloqueo se preserve en ambos niveles.

Entre las técnicas precursoras de control supervisor jerárquico-modular destaca la *voca-*

lización [11] que se basa en la definición de eventos significativos denominados vocales, los cuales describen una abstracción del sistema. Posteriormente se desarrollaron más técnicas con enfoques distintos como las basadas en *interfaces* [15], [16]. También se han propuesto metodologías orientadas a sistemas automatizados de manufactura, donde destaca la técnica de *Control por eventos imperativos* (basada en proyecciones naturales) [17], [18]. Estas técnicas serán explicadas con mayores detalles en el grueso de este trabajo.

Los *sistemas automatizados de manufactura* (SAM) han experimentado un notorio incremento en la complejidad procedural. Además, las condiciones de mercado y las exigencias de los clientes han llevado a incrementar la producción y los requerimientos operacionales relacionados con condiciones ambientales, de seguridad y economía. En el contexto de la investigación los SAM son tomados como SED's, en este sentido, el control de SED's tiene un rol de importancia creciente.

Regularmente, las reglas de producción para manufacturar un producto en específico se suelen incluir como especificaciones en la síntesis del supervisor, lo que da como resultado una arquitectura rígida de supervisión. Sin embargo, es muy común que se tengan que cambiar las reglas de producción en la práctica, consecuentemente en muchas ocasiones se requiere actualizar los supervisores. De esta manera, separar las actividades de control relacionadas con el equipo, de las actividades de manufactura del producto, mediante el uso de técnicas de supervisión jerárquico-modulares, pueden lograr arquitecturas flexibles, además de reducir la complejidad asociada con la síntesis de supervisores y darle la vuelta al problema de la explosión de estados.

En este trabajo de tesis se realiza un estudio de dos técnicas de control jerárquico-modular (*vocalización* y *control por eventos imperativos*), resaltando las ventajas e inconvenientes de cada una en aplicaciones de manufactura así como sus contribuciones frente a otras metodologías. Se presenta el diseño e implementación de las diferentes estructuras de control jerárquicas y modulares aplicadas en un prototipo de SAM hecho en Lego®que cuenta con una interfaz electrónica destinada al acondicionamiento de señales y a la comunicación entre el prototipo y el sistema de control (PC y PLC Siemens Simatic S7-300).

Se propone un esquema de traducción para llevar a la implementación los supervisores sintetizados con cada técnica. Aunque existen otras propuestas de implementación de supervisores [19] y herramientas de generación automática de código [20], [21], [22], [23]; ninguna de éstas corresponden a implementaciones jerárquicas ni los supervisores son empleados como dispositivos de control.

En este trabajo, los supervisores son implementados como controladores, en donde el supervisor puede ejecutar transiciones controlables. En contraste con la TCS, donde el supervisor habilita el conjunto máximo de transiciones controlables permitidas en un estado (i.e. no hay una selección específica sobre cuál transición ejecutar). Los supervisores se traducen de manera ad-hoc dependiendo de la técnica con la que fueron sintetizados.

1.1. Planteamiento del problema

Existen diversas técnicas que logran arquitecturas de tipo jerárquico-modular, cada una propone enfoques diferentes y hace alguna contribución que la distingue de las demás. Todas garantizan la generación de estructuras consistentes, preservando la controlabilidad, el no bloqueo entre los módulos verticales (niveles de jerarquía) y el no conflicto entre los módulos horizontales (módulos que se encuentran en el mismo nivel de jerarquía). Sin embargo, algunas metodologías son más complejas, implican un mayor uso de recursos, difícilmente se pueden llevar a la práctica o conllevan una implementación complicada o que es puede ser muy engorrosa su automatización.

En este trabajo se presentan dos de las técnicas de supervisión jerárquica, *vocalización* y *control por eventos imperativos*, desarrollando formalmente el sistema de supervisión con cada una (síntesis e implementación), con el objetivo de determinar cuál de ellas se adecúa mejor en aplicaciones de manufactura.

1.2. Objetivos

Los objetivos y metas de esta investigación son:

- Realizar un estudio sobre las principales técnicas de control supervisor jerárquico-modular: *Vocalización*, *control por eventos imperativos* e *interfaces*.
- Proponer una metodología de implementación de supervisores jerárquicos.
- Habilitar el prototipo de Sistema de Manufactura construido en Lego®e integrarlo con el equipo de control para la ejecución de pruebas.
- Aplicar el control jerárquico en el prototipo, empleando Vocalización y Control por eventos imperativos; desarrollando formalmente la síntesis de los supervisores y su implementación en un PLC.
- Hacer un análisis comparativo entre las principales técnicas, determinando cuál presenta mayores ventajas en aplicaciones de *control de sistemas de manufactura*.

1.3. Trabajos Relacionados

Lego®Mindstorms® es un juego de robótica fabricado por la empresa Lego®, el cual ha sido ampliamente empleado para la enseñanza y proyectos de investigación en diversas áreas, demostrando su versatilidad y costeabilidad [25], [26].

El Sistema Automatizado de Manufactura (SAM) construido con Lego®está basado en un prototipo de laboratorio y fue diseñado como una cama de pruebas de bajo costo y de

fácil transporte con propósitos de investigación y de enseñanza. El diseño y construcción del prototipo se expone en [27], [28] y [29].

En otros trabajos de tesis se realizó el control del SAM Lego®mediante Simulink [30], empleando bibliotecas desarrolladas en [31] permitiendo la construcción y el análisis de la estructura de control jerárquico-modular de forma gráfica. Posteriormente se exploró el uso de un PLC [32] como dispositivo de control, demostrando tener mayores ventajas en cuanto a flexibilidad y facilidad de uso.

En este trabajo, en cuanto al SAM Lego®, se hace la integración de los trabajos mencionados, desarrollando otra versión de interfaz electrónica basada en [29], desarrollando la capa de entradas-salidas (capa I/O por sus siglas en inglés) para el control del prototipo mediante un PLC de Siemens. A la capa I/O también se le refiere como capa de comunicación entre la planta y el sistema de control, corresponde a la implementación de los componentes elementales del sistema a controlar.

1.4. Organización del documento

El presente trabajo de tesis se encuentra organizado de la siguiente manera. En el *Capítulo 2* se presentan los fundamentos teóricos, incluyendo conceptos básicos de Sistemas de Eventos Discretos y la síntesis de supervisores bajo la Teoría de Control Supervisor. Posteriormente se presenta un resumen sobre la norma ISA-S88 e ISA-S95, definiendo los modelos y terminología aceptada en los procesos industriales y sistemas de manufactura, con la finalidad de estandarizar la nomenclatura y la estrategia de diseño de arquitecturas de control. Se empleará la terminología definida en este estándar a lo largo del documento. Finalmente se encuentra un resumen de las bases teóricas de las principales técnicas de control supervisor jerárquico-modular: *control por eventos imperativos*, *vocalización* e *interfaces*.

En el *Capítulo 3* se plantea un esquema de implementación formal de supervisores. Primeramente se muestra una forma general de implantar la capa I/O (capa de comunicación) entre el sistema a controlar y los supervisores, se expone una manera de representar las máquinas de estados finitos que modelan a los componentes elementales del sistema en lenguaje de escalera. Posteriormente se presenta una traducción a código implementable de los supervisores representados como autómatas, esta traducción es diferente dependiendo de la técnica de control jerárquico empleada. Para el caso de *Vocalización* se establece una traducción formal a diagrama de escalera y en el caso de la técnica de *Control por Eventos Imperativos* se hace una traducción a lenguaje estructurado. Se eligió de esta manera solamente para facilitar la explicación, ya que los supervisores *vocalizados* tienen una estructura más compleja, se prefirió presentarlos en diagrama de escalera para que fuera más clara su visualización; sin embargo pudo haberse escogido de manera indistinta. Se emplearon los dos lenguajes más utilizados en la programación de PLC's, apoyados por el estándar IEC 61131-3. Al exponer la lógica empleada en la traducción de supervisores

en dos lenguajes, mostrando los elementos empleados de cada lenguaje, es posible vislumbrar su implementación en cualquiera de los dos. Para el subconjunto empleado del léxico de cada lenguaje, existe una traducción directa, elemento por elemento, entre lenguaje estructurado y diagrama de escalera. Al presentar los esquemas de traducción de los supervisores, se emplea la terminología definida en la norma ISA-S88 para facilitar referirse a los diferentes niveles de supervisión, sin embargo la implementación no está enfocada únicamente a sistemas de manufactura.

En el *Capítulo 4* se presenta la descripción del prototipo de Sistema Automatizado de Manufactura que se empleó como cama de pruebas para la aplicación de las diferentes técnicas de control supervisor jerárquico-modular. Se presentan los elementos que conforman al sistema y su integración con el equipo de control (PLC Siemens Simatic S7-300, PC e interfaz electrónica).

El *Capítulo 5* muestra la aplicación del control jerárquico, empleando cada una de las técnicas (*control por eventos imperativos* y *vocalización*) en el prototipo mencionado, presentando el diseño de especificaciones y la síntesis de supervisores.

El *Capítulo 6* expone un análisis comparativo entre la técnica de *vocalización* y la de *control por eventos imperativos* en aplicaciones de manufactura. Finalmente en el *Capítulo 7* se encuentran las conclusiones y el trabajo a futuro.

Capítulo 2

Antecedentes Teóricos

En esta sección se muestran los conceptos básicos empleados en Sistemas de Eventos Discretos (SED), una breve introducción a la Teoría de Control Supervisor (TCS) y los fundamentos teóricos relacionados con algunas técnicas de supervisión jerárquica-modular: *control por eventos imperativos*, *vocalización* e *interfaces*. También se presenta un breve resumen sobre la norma IEC 1131 y el uso de dispositivos lógicos programables.

2.1. Sistemas de Eventos Discretos (SED)

Wonham y colaboradores [33], [34], proponen un marco de trabajo para el control de SED basado en Autómatas Finitos Deterministas (AFD) también conocidos como Máquinas de Estados Finitos (MEF). Las secciones 2.1.1 y 2.1.2 son una traducción comentada de [11].

2.1.1. Máquina de Estados Finitos (MEF)

Un Autómata Finito (AF) o Máquina de Estados Finitos (MEF) es un modelo matemático que realiza cómputos en forma automática sobre una entrada para producir una salida. La estructura formal de una MEF está dada por:

$$G := (Q, \Sigma, \delta, q_o, Q_m)$$

Donde:

- Q es el conjunto de estados.
- Σ es el alfabeto finito de símbolos a las que se les refiere como etiquetas de eventos o transiciones.

  $\delta : Q \times \Sigma \rightarrow Q$ es la función parcial de transición.

  q_0 es el estado inicial.

- $Q_m \subseteq Q$ es el conjunto de estados marcados.

Un evento de G es una tripleta $(q, \sigma, \tilde{q})$ donde $\delta(q, \sigma) = \tilde{q} \quad q, \tilde{q} \in Q$. En esta representación q es el estado de salida, $\tilde{q}$ es el estado de entrada y $\sigma \in \Sigma$ es la etiqueta del evento que lleva al estado de entrada a partir del estado de salida.

El alfabeto Σ está formado por todas las transiciones controlables e incontrolables $\Sigma = \Sigma_c \bigcup \Sigma_u$. En un grafo de estado-transición el estado inicial es etiquetado con una flecha entrante, mientras que una flecha saliente representará a un estado marcado. Un estado inicial y marcado se representa con una doble flecha.

2.1.2. Control Supervisor

En Teoría de Control Supervisor, tanto el modelo del sistema (planta) como el comportamiento requerido (especificación) se encuentran representados por autómatas. Se sintetizan supervisores, también representados con autómatas cuyo lenguaje de aceptación es un sublenguaje controlable del sistema. Los supervisores sintetizados restringen el comportamiento de la planta de manera que cumplan, en la medida de lo posible, las especificaciones de comportamiento.

Se asume que la planta genera espontáneamente los eventos. El alfabeto de entrada Σ del autómata está relacionado con los eventos que pueden suceder en el SED se divide en controlable e incontrolable, dependiendo de si es posible ejercer acción externa sobre la ocurrencia de un evento o no. El supervisor observa las cadenas de palabras generadas por la planta y puede acotar su comportamiento mediante la deshabilitación de transiciones controlables.

Definición 2.1.1 (**Controlabilidad**). La noción de controlabilidad se captura con una expresión lingüística que implica a un lenguaje cerrado K que representa una especificación y al lenguaje L generado por el autómata G que representa al sistema. Se dice que un lenguaje K es controlable con respecto a L si y sólo si (sii):

$$\bar{K}\Sigma_u \cap L(G) \subseteq \bar{K}$$

donde $\bar{K}$ denota al conjunto de los prefijos de K y Σ_u al conjunto de transiciones incontrolables. Lo cuál se puede interpretar como sigue: "si no es posible evitar la ocurrencia de un evento, entonces su ocurrencia debe ser válida respecto a la especificación que se requiere cumplir"

Definición 2.1.2 (**Lenguaje Cerrado**). Dado un lenguaje $K \subseteq L \subseteq \Sigma^*$ se dice que K es $L - cerrado$ sii:

$$K = \bar{K} \cap L$$

Definición 2.1.3 (**Supervisor**). Un supervisor es un mapeo $V : L(G) \rightarrow \Gamma$ tal que cada componente del patrón de control $\gamma \in \Gamma$ contiene todos los eventos incontrolables habilitados por el sistema y aquellos controlables permitidos por la especificación, atribuyendo al comportamiento de lazo cerrado como controlable. El supervisor V es frecuentemente realizado por la pareja $\mathcal{S} := (S, \phi)$ con la MEF $S = (Y, \Sigma, \delta_S, y_0, Y_m)$ y $\phi : Y \rightarrow \Gamma$. La acción de control es ejercida por $\mathcal{S}(s) = \phi(\delta_S(y_0, s))$ para cada $s \in L(G)$, sincronizando así transiciones incontrolables con señales de entrada y transiciones controlables con comandos de control.

Para indicar que G está bajo la supervisión de V se escribe V/G. El comportamiento del sistema a lazo cerrado es $L(V/G) \subseteq L(G)$ se determina como:

- $\epsilon \in L(V/G)$
- Si $s\sigma \in L(V/G)$, $\sigma \in V(s)$ y $s\sigma \in L(G)$ entonces $s\sigma \in L(V/G)$
- No hay otra cadena que pertenece a $L(V/G)$

El lenguaje marcado de V/G está dado por:

$$L_m(V/G) = L(V/G) \cap L_m(G)$$

Con respecto a G se dice que V es no bloqueante si:

$$L_m(\bar{V}/G) = L(V/G)$$

Teorema 2.1.1. Dado $K \subseteq L_m(G)$, $K \neq \emptyset$. Existe un supervisor no bloqueante V para G tal que $L_m(V/G) = K$ sii:

- K es controlable con respecto a G y
- K es $L_m(G) - cerrado$

Corolario 2.1.1. Dado $K \subseteq L_m(G)$, $K \neq \emptyset$. Existe un supervisor marcado y no bloqueante V para (K, G) tal que:

$$L_m(V/G) = K$$

Si y solo si K es controlable con respecto a G. Esta es una simplificación del teorema anterior puesto que la condición $L_m(G) - cerrado$ ahora se puede quitar.

Síntesis de Supervisores

Con los conceptos presentados anteriormente se establece la existencia de un control supervisor no bloqueante para un SED dado.

Sea un SED con $\Sigma = \Sigma_c \cup \Sigma_u$, sea $E \subseteq \Sigma^*$; donde E será el lenguaje de especificación para el control supervisor de G. Se introduce el conjunto de todos los sublenguajes de E que son controlables con respecto a G:

$$C(E) = \{K \subseteq E \mid K \text{ controlable con respecto a } G\}$$

$C(E)$ es no vacío y es cerrado bajo uniones arbitrarias. En particular, $C(E)$ contiene un único elemento supremo:

$$supC(E) = \bigcup \{K \mid K \in C(E)\}$$

Teorema 2.1.2. Sea $E \subseteq \Sigma^*$ y $K = supC(E \cap L_m(G))$. Si $K \neq \emptyset$, entonces existe un supervisor no bloqueante V para G tal que $L_m(V/G) = K$.

El supervisor es el supremo sublenguaje controlable que cumple con todas las especificaciones planteadas para el sistema y que, como su nombre lo indica, produce un lenguaje controlable para restringir el comportamiento del sistema.

Tanto el sistema como las especificaciones son modeladas por MEF's, se sincronizan todas las MEF's que corresponden a los componentes elementales del sistema y comportamientos causales para obtener el modelo de la planta; así mismo, las especificaciones individuales (de seguridad, procedimiento o compartición de recursos) se sincronizan para obtener una especificación monolítica. Posteriormente se hace una sincronización de las MEF's que corresponden a la planta y la especificación monolítica, sobre la cual se calcula el supremo controlable que resultará en el control supervisor del sistema.

La síntesis de los supervisores presentada en este trabajo fue realizada con TCT [11] y Supremica [35]. TCT es un paquete de software de dominio público desarrollado por Karen Rudie, Pablo Iglesias, Jimmy Wong y Pok Lee; el cual contiene diversos algoritmos relacionados con sistemas de eventos discretos no temporizados y la síntesis de supervisores. Supremica, al igual que TCT, es una herramienta de software para la síntesis y verificación formal de sistemas de control de eventos discretos, este software se puede descargar gratuitamente para propósitos educativos, de investigación o no lucrativos.

2.2. La norma ISA-S88

El estándar ISA-S88 ha sido ampliamente usado por el mercado de la automatización de los procesos de fabricación por lotes, que permiten la producción flexible sin necesidad de re-ingeniería cuando se requiere introducir mejoras en los procesos o introducir nuevos

productos. Esta sección es un resumen de [38], presentando brevemente el contenido de las cuatro partes de la norma ISA-S88 así como la terminología empleada.

La norma ISA-S88 Parte 1 (ANSI-ISA-88.01-1995) define los *modelos* y *terminología* que deben aplicar los sistemas de control por lotes. Gracias a la aparición de esta parte de la norma en el año 1995, pudieron establecerse modelos y términos que han permitido que todos los actores que participan en la industria del control secuencial o por lotes (*Batch Control* en inglés) unifiquen su lenguaje, independientemente de las herramientas utilizadas en su implementación. Así, actualmente puede hablarse de términos como *celda de proceso*, *unidad*, *módulos de equipo* o *módulos de control* de una forma inequívoca, en lo que al *Modelo Físico* se refiere; o bien, de *procedimientos*, *procedimientos unitarios*, *operaciones* y *fases* incluidos en el *Modelo de Control de Procedimientos* (la Figura 2.1 muestra ambos modelos).

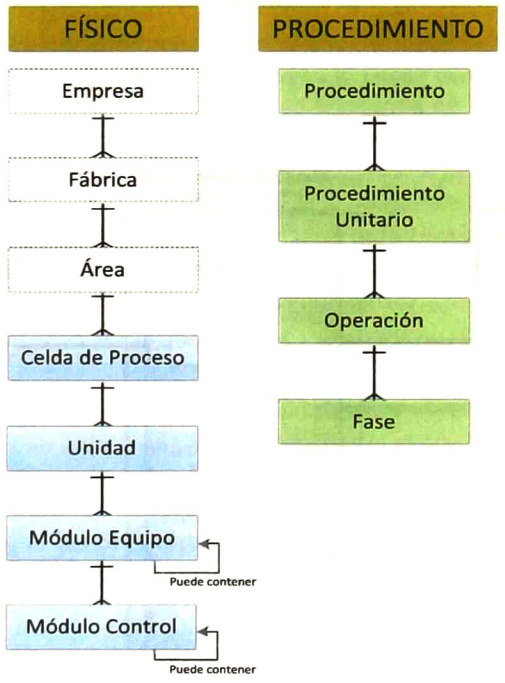


Figura 2.1: Modelo Físico y de Procedimientos ISA-S88

La Parte 1 de la norma también define los tipos de *recetas* que deben gestionar las Compañías con el fin de obtener máximos niveles de flexibilidad e integración entre sus sistemas corporativos y de planta, y entre diferentes fábricas u organizaciones. Así, se definen los siguientes tipos de recetas: *receta general*, *receta de fábrica*, *receta maestra* y *receta de control*. Mientras que las recetas *general* y de *fábrica* describen los

procesos de fabricación para la producción independientemente de los requerimientos de equipos; las recetas *maestra* y de *control* describen las acciones específicas requeridas en determinados equipos para producir un lote del producto. Las Partes 2 y 3 del estándar han proporcionado más detalles acerca de las estructuras de datos y lenguajes (Parte 2) y de la implementación de la receta *general* y de *fábrica* (Parte 3). La Parte 4 del estándar define el modelo de referencia para los registros de producción de lote.

En la Figura 2.2 se presenta el modelo de referencia de la arquitectura para las actividades realizadas en el nivel de coordinación de recursos de sistemas por lotes, continuos y discretos. El *Modelo Físico* es un agrupamiento jerárquico de los activos físicos. El resultado de combinar los equipos con los procedimientos es llamado *modelo de proceso.* El *Modelo de Control de Procedimientos* establece las actividades que ejecutan los activos físicos para realizar tareas con un sentido de proceso. Mediante la combinación de los *Modelos Físico* y de *Procedimiento* es posible definir la secuencia de actividades químicas, físicas o biológicas para la fabricación de una cantidad finita de material (lote), en una secuencia de *procedimientos de unidad, operaciones* y *fases* a ejecutar en una *celda de proceso.*

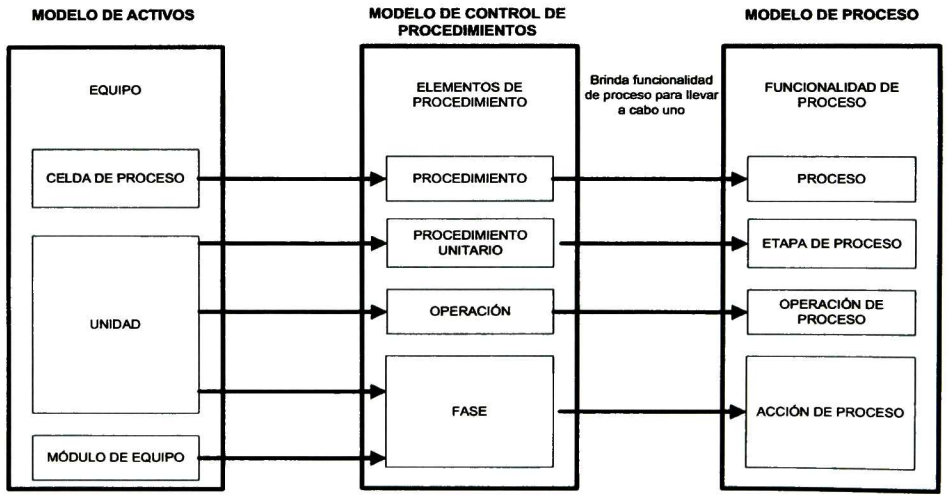


Figura 2.2: Modelo de Referencia de la Arquitectura

En este trabajo se le refiere como *procedimiento de receta* o simplemente *receta* al "plan de producción" que se encuentra en el nivel de mayor jerarquía de la arquitectura de control, estableciendo una secuencia ordenada de ejecución de tareas (comandos para realizar un procedimiento o completar la manufactura de un producto específico).

2.3. Técnicas de Control Jerárquico-Modular

La sección 2.3.1 es un resumen comentado de [17] y [18] el cual presenta las bases teóricas sobre la técnica de *control por eventos imperativos*. La sección 2.3.2, que se refiere a los fundamentos teóricos de la *vocalización*, es una traducción comentada de [11].

2.3.1. Control por Eventos Imperativos

Las reglas de producción para manufacturar un producto en específico se suelen incluir como especificaciones en la síntesis del supervisor, lo que da como resultado una arquitectura rígida de supervisión. Sin embargo, es muy común que se tengan que cambiar las reglas de producción en la práctica (introducción de nuevos productos o cambios en el proceso), consecuentemente se requiere actualizar los supervisores. De esta manera, separar las actividades de control relacionadas con el equipo de las acciones de manufactura del producto puede ayudar a diseñar mejores arquitecturas de control supervisor. Un paso en esta dirección es la síntesis de supervisores estableciendo rutas viables de manufactura para productos específicos (supervisores de producto) en términos de los modelos de tareas del equipo. Estos supervisores se relacionan con los supervisores de la estación de trabajo por medio del conjunto de transiciones vinculadas con tareas de equipo que tienen en común. Este esquema da lugar a una arquitectura de tipo jerárquico-modular, en donde la síntesis de los supervisores de producto del nivel de mayor jerarquía requiere de proyecciones sobre las especificaciones o supervisores del nivel bajo relacionados con el comportamiento del equipo. Por lo tanto, hacer cambios en los supervisores del nivel bajo puede implicar la reformulación de los supervisores de producto.

La aplicación de esta metodología está orientada a una clase de sistemas automatizados de manufactura en los que existe una clara separación entre las actividades de control del equipo y los procedimientos de manufactura de productos. Se sintetizan supervisores de equipo encargados únicamente con cuestiones de operación y seguridad de las agrupaciones de equipo mientras que los procedimientos genéricos de manufactura (denominados *procedimientos de receta*) se modelan con lenguajes regulares construidos con una clase de comandos de control denominados *imperativos*.

El control de los módulos de equipo se lleva a cabo siguiendo las secuencias de los comandos *imperativos* de control. Se establecen condiciones para garantizar la manufacturabilidad de un producto dado un supervisor sintetizado para un SAM particular, de esta manera no es necesario modificar los controladores de equipo al considerar diferentes productos a manufacturar.

Sistema Confiable y Lenguaje Realizable

Esta sección es una traducción de [18], solamente haciendo una corrección en la definición de autómata confiable; para mayores detalles consultar [17].

La construcción de un control de supervisión que garantice la fabricación de un producto dado está basada en la noción de un sistema confiable y lenguaje realizable.

A causa de los conceptos de sistema confiable y lenguaje realizable $\Sigma = \Sigma_c \cup \Sigma_u$ se clasifica en eventos imperativos $\Sigma_{imp} \subseteq \Sigma_c$ los cuales son eventos controlables que describen el inicio de una tarea o proceso. El resto de los eventos son llamados informativos, $\Sigma_{inf} \subseteq \Sigma$ donde $\Sigma_{imp} \cap \Sigma_{inf} = \emptyset$.

Definición 2.3.1 (**Estado de Decisión**). Dado una MEF G, un estado $q \in Q$ se dice que es *un estado de decisión* si para cualquier evento $\sigma \in \Sigma$ tal que $\delta(q, \sigma)!$ se cumple que $\sigma \in \Sigma_c$ y existe al menos una transición $\rho \in \Sigma_{imp}$ tal que $\delta(q, \rho)!$. Es decir, un estado $q \in Q$ es de decisión si todos los eventos habilitados en él son controlables y al menos uno es imperativo.

Definición 2.3.2 (**Estado Lograble**). Para una MEF G dada, un estado $\tilde{q} \in Q$ se dice que es un estado *lograble* de $q \in Q$ y se denota como $q - \rightarrow \tilde{q}$ si existe $s \in \Sigma_{inf}^*$ tal que $\delta(q, s) = \tilde{q}$. Un estado es lograble desde otro estado si existe un camino del estado de origen al estado lograble formado sólo por eventos informativos.

Definición 2.3.3 (**Autómata Confiable**). Un autómata G se dice que es *confiable* si los eventos *imperativos* $\sigma \in \Sigma_{imp}$ satisfacen que para todo $q \in Q$ con $\delta(q, \sigma)$ definida, existe un estado de decisión $\tilde{q}$ lograble a partir de $\delta(q, \sigma)$, es decir $\delta(q, \sigma) - \rightarrow \tilde{q}$.

Esta definición fue modificada a como aparece en el texto original debido a que contenía un ligero error: se establecía que el estado de decisión $\tilde{q}$ es lograble a partir de q; sin embargo en el estado q se encuentra definida la función $\delta(q, \sigma)$ donde $\sigma \in \Sigma_{imp}$, es decir q tiene al menos una transición imperativa habilitada, que por definición es controlable. Entonces no puede ser que $q - \rightarrow \tilde{q}$, ya que por la definición de lograble, el camino que une a q con $\tilde{q}$ está formado únicamente por transiciones incontrolables.

En un autómata confiable todos los eventos imperativos pueden eventualmente ser ejecutados. El algoritmo que se muestra enseguida verifica si un autómata G es confiable.

Sea $f : \Sigma_{imp} \rightarrow 2^Q$ donde $f(\sigma)$ es un conjunto de estados de decisión.

Sea $g : \Sigma_{imp} \rightarrow 2^Q$ donde $g(\sigma)$ es un conjunto de estados, donde existen dos transiciones $\sigma \in \Sigma_{imp}$ y $\sigma" \in \Sigma_u$ tal que $\delta(q, \sigma)$ y $\delta(q, \sigma")$ están definidas.

For $\sigma \in \Sigma_{imp}$ ***do***
    If $f(\sigma) = \emptyset$ ***then***

G no es confiable, finalizar
Else
If $f(\sigma) \neq \emptyset$ *y* $g(\sigma) \neq \emptyset$ ***then***
Eliminar todos los eventos imperativos en G
If $g(\sigma) \not\subseteq C_o(f(\sigma))$ ***then***
G no es confiable, finalizar
Next
Return G es confiable

$C_o(f)$ es el conjunto coalcanzable de f obtenido de forma iterativa como la pre-imagen f. Es fácil ver que el producto asíncrono de dos MEF's confiables es también una MEF confiable.

Sea $M := (Q, \Sigma, \delta, q_o, Q_m)$ una MEF cuyo lenguaje describe el conjunto de secuencias de comandos de fabricación para un producto en términos de eventos imperativos.

Definición 2.3.4 (**Lenguaje Realizable**). Para las MEF's G y M, el lenguaje $L(M)$ es realizable en G si se cumple:

1. $L(M) \subset Proj(L(G))$ donde $Proj : \Sigma^* \rightarrow \Sigma^*_{imp}$
2. Si $w_i \in L(G)$ y $w_i \in Proj^{-1}(s_i)$ con $s_i \notin L_m(M)$, entonces existe un estado de decisión $q_i = \delta_G(q_0, z_i)$ tal que $\delta_G(q_0, z_i) - \rightarrow q_i$ y $Proj(z_i)\xi \in L(M)$ donde ξ es un evento que está habilitado en q_i.

Si el lenguaje $L(M)$ es realizable sobre la MEF a lazo cerrado V/G entonces todas las secuencias de fabricación en $L(M)$ pueden ser seguidas evento por evento en la MEF a lazo cerrado V/G y así todos los eventos imperativos pueden ser ejecutados de una manera confiable.

Proposición 2.3.1. Dado una MEF confiable G, el lenguaje $L(M)$ es realizable en G si $Proj(L(M) \parallel L(G))$ es isomorfo a $L(M)$.

2.3.2. Vocalización

En esta sección se presenta la técnica de vocalización de manera general, para más detalles consultar el capítulo 5 de [11] y [14].

La *vocalización* es una técnica que permite lograr un esquema de supervisión jerárquica de sistemas de eventos discretos mediante la definición de eventos significativos denominados vocales los cuales describen una abstracción del sistema.

Se considera una estructura de control jerárquico de dos niveles (mostrada en la Figura 2.3) formada por los siguientes componentes: G_{lo} es la planta que representa el

comportamiento del sistema en el nivel inferior, C_{lo} es el controlador (representado por una función de mapeo) en el nivel inferior, G_{hi} es la planta del nivel superior (es una abstracción de G_{lo}), C_{hi} es el supervisor del nivel superior, Inf_{lohi} es el canal de información que permite actualizar a G_{hi} con respecto a G_{lo}.

Inf_{lo} e Inf_{hi} son canales de información cuya función es retroalimentar a C_{lo} y C_{hi} con respecto de G_{lo} y G_{hi}; Con_{lo} y Con_{hi} son canales de control que permiten aplicar el control a G_{lo} y G_{hi}; Com_{hilo} es el canal de comando que permite enviar la señal de control como un comando a C_{lo}, éste a su vez tiene que enviar ese comando por el canal de control Con_{lo} para aplicarlo en G_{lo}.

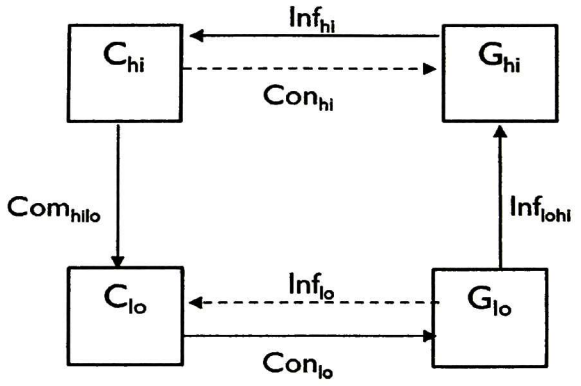


Figura 2.3: Control jerárquico de dos niveles

El control ejercido por C_{hi} es "virtual" en el sentido de que el comportamiento de G_{hi} está determinado por el comportamiento de G_{lo} a través del proceso de refresco de Inf_{lohi}

Sea T un conjunto no vacío de eventos significativos (*vocales*), percibidos por el nivel administrador y por lo tanto describen G_{hi}. Para modelar el canal de información, se tiene el mapeo:

$$\theta : L_{lo} \rightarrow T^*$$

con las propiedades:

$$\begin{aligned} \theta(\epsilon) &= \epsilon \\ \theta(s\sigma) &= \begin{cases} \theta(s) \\ \theta(s)\tau, \tau \in T \end{cases} \end{aligned}$$

Para $s \in L_{lo}$, $\sigma \in \Sigma$. θ puede ser usado para señalar la ocurrencia de un evento que depende de alguna manera del pasado histórico del comportamiento de G_{lo}. Se combina θ con G_{lo} en una sola descripción, lo cual se logra reemplazando el par (G_{lo}, θ) por el generador de Moore con alfabeto de salida: $T_o = T \cup \{\tau_o\}$, donde τ_o es un nuevo símbolo ($\tau_o \notin T$) interpretado como "símbolo de salida silenciosa" Con este objetivo, se escribe temporalmente:

$$\bar{G}_{lo} = (\bar{Q}, \Sigma, T_o, \bar{\delta}, \omega, \bar{q}_o, \bar{Q}_m)$$

Donde los elementos con barra juegan el mismo papel que en G_{lo}, mientras que $\omega : \bar{Q} \to T_o$ denota el mapeo de los estados y su símbolo de salida correspondiente. Para ω se define:

$$\omega(\bar{q}_o) = \tau_o$$

Si $\bar{\delta}(\bar{q}_o, s\sigma)!$ entonces:

$$\begin{aligned} \omega\left(\bar{\delta}(\bar{q}_o, s\sigma)\right) &= \tau_o \text{ si } \theta(s\sigma) = \theta(s) \\ \omega\left(\bar{\delta}(\bar{q}_o, s\sigma)\right) &= \tau \text{ si } \theta(s\sigma) = \theta(s)\tau \end{aligned}$$

Es decir, ω entrega el símbolo silencioso τ_o si θ entrega nada nuevo y en caso contrario ω entrega el nuevo símbolo $\tau \in T$

Por ahora, G_{hi} se considerará como el reconocedor canónico para la imagen de L_{lo} bajo θ :

$$L(G_{hi}) = \theta(L_{lo}) \subseteq T^*$$

y escribiremos $L(G_{hi}) := L_{hi}$. Ahora, se debe refinar la estructura de estados de G_{lo} para extender el alfabeto del nivel superior T y particionar la extensión en subconjuntos controlables e incontrolables. Considerando el árbol de alcanzabilidad de L_{lo} con estado inicial $q = 0$ como la raíz. Cada nodo del árbol es etiquetado con el valor correspondiente $\tau' \in T_o$. Se define $\hat{\omega} : N \to T_o$ donde N es el conjunto de nodos del árbol de alcanzabilidad de L_{lo}. En el árbol, los nodos con $\tau' = \tau_o$ son *silenciosos* y aquellos con $\tau' \in T$ son *vocales*. Un *camino silencioso* en el árbol es aquel que une dos nodos vocales o la raíz con un nodo vocal, donde todos los nodos intermedios (si es que los hay) son silenciosos. Esquemáticamente, un *camino silencioso* tiene la forma:

$$n \xrightarrow{\sigma} s \xrightarrow{\sigma'} s' \longrightarrow \ldots \longrightarrow s'' \xrightarrow{\sigma''} n'$$

donde el nodo inicial n es vocal o es el nodo raíz y los nodos silenciosos intermedios $s, s', ..., s''$ pueden estar ausentes. Un camino silencioso será *rojo* si al menos una de las transiciones $\sigma \in \Sigma$ es controlable, en caso contrario se le denominará que es color *verde*. Se colorea el nodo vocal de acuerdo al color del camino silencioso y se crea el alfabeto extendido T_{ext}. Se define $\hat{\omega}_{ext} : N \to T_{ext}$ de acuerdo con:

$$\begin{aligned} \hat{\omega}_{ext}(n) &= \tau_o \text{ si } n \text{ es silencioso} \\ \hat{\omega}_{ext}(n) &= \tau_c \text{ si } \hat{\omega}(n) = \tau \in T \text{ y color}(n) = rojo \\ \hat{\omega}_{ext}(n) &= \tau_i \text{ si } \hat{\omega}(n) = \tau \in T \text{ y color}(n) = verde \end{aligned}$$

Es claro que $\hat{\omega}_{ext}$ implica la extensión $\theta_{ext} : L_{lo} \to T_{ext}^*$. Finalmente, se define:

$$G_{lo,ext} = (Q_{ext}, \Sigma, T_{ext}, \delta_{ext}, \omega_{ext}, q_o, Q_{m,ext})$$

Por construcción, es fácil ver que $|T_{ext}| \leq 2|T| + 1$: el número de símbolos de salida no-silenciosos a lo mucho se ha duplicado al clasificar los antiguos símbolos de salida en

controlables e incontrolables. Se dice que $G_{lo,ext}$ es **consistente en salidas de control (OCC por sus siglas en inglés).** El algoritmo *outconsis* de TCT hace a $\bar{G}_{lo}$ consistente en salidas de control. La estructura correspondiente G_{hi} se obtiene mediante el algoritmo *higen* incorporado a TCT.

C_{hi} se define por medio de un mapeo:

$$\gamma_{hi} : L_{hi} \times T \rightarrow \{0, 1\}$$

Donde:

$$\gamma_{hi}(t, \tau) = 1 \text{ habilitado } t \in L_{hi} \text{ y } \tau \in T_u$$
$$\gamma_{hi}(t, \tau) = 0 \text{ deshabilitado } t \in L_{hi} \text{ y } \tau \in T_c$$

El resultado de aplicar directamente este control en la acción generada por G_{hi} sintetizaría el lenguaje de lazo cerrado:

$$L(\gamma_{hi}, G_{hi}) \subseteq T^*$$

Se define el mapeo del nivel superior para la deshabilitación de eventos:

$$\Delta_{hi} : L_{hi} \rightarrow Pwr(T_c)$$

Donde:

$$\Delta_{hi}(t) = \{\tau \in T_c | \gamma_{hi}(t, \tau) = 0\}$$

De manera correspondiente, el mapeo del nivel bajo para deshabilitar eventos será:

$$\Delta_{lo} : L_{lo} \times L_{hi} \rightarrow Pwr(\Sigma_c)$$

De acuerdo con:

$$\Delta_{lo}(s, t) = \sigma \in \Sigma_c$$

Tal que se cumple lo siguiente:

1. Existe $s' \in \Sigma_u^*$ tal que $s\sigma s' \in L_{lo}$
2. $\hat{\omega}(s\sigma s') \in \Delta_{hi}(t)$
3. $(\forall s'')\, s'' < s' \Rightarrow \hat{\omega}(s\sigma s'') = \tau_o$

Cuando se cierra el lazo mediante el canal de información Inf_{lohi}, una cadena $s \in L_{lo}$ es mapeada por $t = \theta(s) \in L_{hi}$. El control implementado por C_{lo} está dado por:

$$\gamma_{lo}(s, \sigma) = \begin{cases} 0 \text{ si } \sigma \in \Delta_{lo}(s, \theta(s)) \\ 1 \text{ en otro caso} \end{cases}$$

Ahora suponemos un lenguaje no vacío cerrado "legal" o de especificación $E_{hi} \subseteq L_{hi}$ para el controlador de alto nivel C_{hi}. Se asume que E_{hi} es controlable con respecto al modelo estructural del nivel superior:

$$E_{hi}T_u \cap L_{hi} \subseteq E_{hi}$$

E_{hi} puede ser sintetizado como el comportamiento controlable de G_{hi} bajo la ley de control γ_{hi}. En la teoría estándar la determinación de γ_{hi} no es usualmente única, sin embargo γ_{hi} debe siempre satisfacer:

$$\gamma_{hi}(t,\tau) = 0 \quad si \quad t \in E_{hi} \;\; y \;\; t\tau \in L_{hi} - E_{hi}$$

Se define a E_{lo} como el (máximo) comportamiento de G_{lo} que será transmitido por Inf_{lohi} como comportamiento E_{hi} en el modelo de nivel superior G_{hi}:

$$E_{lo} = \theta^{-1}(E_{hi}) \subseteq L_{lo}$$

Donde $E_{hi} = \theta(E_{lo})$ y $L_{hi} = \theta(L_{lo})$. Claramente E_{lo} es cerrado pero en general no será cierto que E_{lo} es controlable con respecto a G_{lo}.

Teorema 2.3.1. Bajo las suposiciones anteriores:

$$L(\gamma_{lo}, G_{lo}) = E_{lo}^{\uparrow}$$

La condición establecida en el Teorema 2.3.1 se le llama *consistencia jerárquica de nivel bajo*. Esta propiedad garantiza que la actualización del comportamiento de G_{hi} siempre satisface la restricción del nivel superior, y que el comportamiento real del nivel bajo en G_{lo} puede ser tan amplio como sea posible mientras satisfaga la restricción; sin embargo, el comportamiento esperado por el administrador C_{hi} sobre G_{hi} puede ser más amplio de lo que el operador C_{lo} sobre G_{lo} pueda reportar.

Bajo las suposiciones actuales sobre G_{lo}, una llamada de C_{hi} para la deshabilitación de algún evento de nivel superior $\tau \in T_c$ puede requerir que C_{lo} (el control γ_{lo}) deshabilite caminos en G_{lo} que guíen directamente hacia salidas diferentes a τ, por lo que es necesario imponer más condiciones sobre G_{lo}.

Sea $E_{hi} \subseteq L_{hi}$ cerrado y controlable y sea G_{lo} *consistente en salidas de control*, donde la inclusión:

$$\theta\left(\left(\theta^{-1}(E_{hi})\right)^{\uparrow}\right) \subseteq E_{hi}$$

puede ser estricta ($\subset$); el comportamiento E_{hi} esperado por el administrador en G_{hi} puede ser más amplio de lo que el operador pueda notarlo: el administrador es "demasiado optimista" con respecto a la eficacia del proceso de comando de control.

Si se preserva la igualdad en:

$$\theta\left(\left(\theta^{-1}(E_{hi})\right)^{\uparrow}\right) \subseteq E_{hi}$$

Entonces el par (G_{lo}, G_{hi}) tendrá *consistencia jerárquica*, en este caso, por el Teorema 2.3.1 el proceso de comando de control en E_{hi} sintetizará E_{hi} en G_{hi}.

Para obtener la igualdad en la expresión anterior para una especificación controlable con lenguaje E_{hi} se requiere de un refinamiento adicional en la estructura de las transiciones de G_{lo}, en otras palabras, mejorar la información que manda C_{lo} a C_{hi} (o de G_{lo} a G_{hi} según la interpretación).

Refiriéndose al árbol de alcanzabilidad de $G_{lo,ext}$ se dirá que los nodos vocales rojos n_1, n_2 con $\tilde{\omega}(n_1) \neq \tilde{\omega}(n_2)$ son *socios* si sus caminos silenciosos comienza ya sea en el nodo raíz o comienza en el mismo nodo vocal.

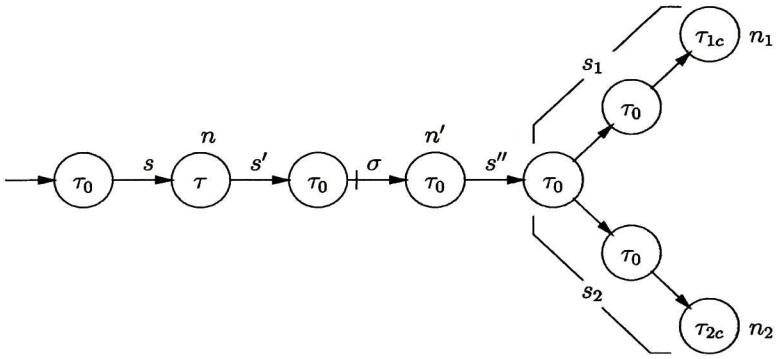


Figura 2.4: Socios n_1, n_2 con antecedente n'

Sea $n = node(s)$, comparten un segmento inicial $s'\sigma$ con $\sigma \in \Sigma_c$, y a este segmento compartido le siguen segmentos etiquetados como $s''s_1$, $s''s_2$ respectivamente, donde $s'' \in \Sigma_u^*$ y al menos una cadena s_1, s_2 pertenece a Σ_u^*. Llamaremos al nodo $(ss'\sigma)$ el antecedente de los nodos socios.

En esta estructura, los eventos controlables $\tau_{1c} = \tilde{\omega}(n_1)$, $\tau_{2c} = \tilde{\omega}(n_2)$ en G_{hi} no podrán ser deshabilitados de manera independiente por un comando de control hacia C_{lo}. Es decir: si E_{hi} requiere que se deshabilite τ_{2c} entonces C_{lo} se verá forzado a deshabilitar τ_{1c} también, deshabilitando directamente la transición controlable σ. Una solución a este problema es dividir o separar la ocurrencia de los socios: declarando el nodo antecedente como nodo vocal rojo con salida controlable τ_c'' (nuevo símbolo que extiende T_c) y hay que re-colorear el árbol.

Definición 2.3.5. Sea G una estructura de Moore, entonces G será *estrictamente consistente en salidas de control* (SOCC por sus siglas en inglés) si es *consistente en salidas de control* y en el árbol de alcanzabilidad de $L(G)$ ninguna pareja de nodos rojos son socios.

Teorema 2.3.2. Asumiendo que G_{lo} es SOCC, y sea $E_{hi} \subseteq L(G_{li})$ no vacío, cerrado y controlable. Entonces:

$$\theta((\theta^{-1}(E_{hi}))^{\uparrow}) = E_{hi}$$

Por lo tanto, cuando G_{lo} es SOCC se logra la consistencia jerárquica para el par (G_{lo}, G_{hi}). En TCT la consistencia jerárquica se logra mediante el algoritmo *hiconsis*, el cual comprueba que en la estructura de Moore introducida no haya nodos socios y en caso de haberlos propone la vocalización de los nodos antecedentes para hacerla SOCC. En ocasiones, la solución propuesta por este algoritmo no es la mejor, existe la alternativa de cambiar la vocalización de los estados para obtener estructuras SOCC más sencillas.

Es importante notar que la restricción de hacer el control jerárquico en dos niveles fue completamente secundario. Una vez que se logra la consistencia jerárquia en el nivel más bajo y el primer nivel hacia arriba, sea (G_o, G_1), la construcción se puede repetir asignando salidas a los estados en G_1 y recurrir a un siguiente nivel superior G_2. La consistencia de (G_1, G_2) se logra sin perturbar la de (G_o, G_1).

Definición 2.3.6 (Condición Principal (MC)). Sea G_{lo} consistente en salidas de control (OCC) y recordando que $C(E)$ denota la familia de sublenguajes controlables de E. La condición principal se define como:

$$\theta C(L_{lo}) = C(L_{hi})$$

Lo que significa que θ no solamente preserva la controlabilidad, sino que también denota que todo lenguaje controlable del nivel alto es la imagen de algún lenguaje controlable de nivel bajo.

Dicho con otras palabras, toda tarea que pueda ser especificada en el modelo del administrador G_{hi} es ejecutable en el modelo del operador G_{lo}; las políticas de alto nivel siempre pueden llevarse a cabo en el bajo nivel.

Teorema 2.3.3.

$$MC \Leftrightarrow \left[(\forall E_{hi})\, E_{hi} \subseteq L_{hi} \Rightarrow \theta((\theta^{-1}(E_{hi}))^{\uparrow}) = E_{hi}^{\uparrow}\right]$$

La utilidad de este teorema radica en el hecho que la condición de consistencia jerárquica que implicaba la operación $(.)^{\uparrow}$ es reemplazada por la condición principal (MC) que es formalmente más sencilla, ya que involucra únicamente la propiedad de controlabilidad.

Teorema 2.3.4.

$$\theta C(L_{lo}) \supseteq C(L_{hi}) \Leftrightarrow \left[(\forall E_{hi})\, E_{hi} \in C(L_{hi}) \Rightarrow \theta((\theta^{-1}(E_{hi}))^{\uparrow}) = E_{hi}\right]$$

Estos resultados dependen del hecho de que las operaciones $\theta(.)$ y $(.)^{\uparrow}$=*sup*C(.) son monótonas en sublenguajes.

2.3.3. Otras técnicas

En las técnicas existentes de control supervisor jerárquico-modular se pueden distinguir dos estrategias diferentes para reducir la complejidad asociada con la síntesis de supervisores:

- Técnicas aplicadas en abstracciones incrementales construidas a partir del sistema o de un módulo del sistema, como es el caso de la *Vocalización* [11].
- Técnicas que se basan en el análisis y diseño local, dividiendo al sistema en módulos más pequeños y limitando o restringiendo la interacción entre los módulos o entre los niveles de supervisión. Generalmente con el objetivo de que las propiedades globales puedan ser verificadas localmente. Estas técnicas generan arquitecturas de supervisión altamente reconfigurables ya que se puede modificar un componente del sistema sin tener que re-analizar el sistema global. La técnica de *Control por eventos imperativos* [17], [18] y la de *Interfaces* [15], [16] son ejemplos de este tipo de enfoque.

Enseguida se expondrá un resumen de la técnica de *Interfaces*, sin embargo no se tratará en este trabajo debido a que no presenta ventajas significativas frente a las técnicas expuestas en las secciones 2.3.1 y 2.3.2. Se abordará únicamente la técnica de *Vocalización* como representante de las metodologías basadas en abstracciones incrementales y la técnica de *Control por eventos imperativos* para representar aquellas basadas en análisis local. Se ha excluido la técnica de *Interfaces* por ser muy similar a la técnica de *Control por eventos imperativos*, dado un sistema se puede llegar a una solución muy similar con esta última técnica, además que ésta ofrece como ventaja el concepto de realizabilidad de la receta y es inherente a sistemas de manufactura. Conjuntamente, se garantizan las propiedades globales al ir construyendo autómatas confiables que pueden ser menos restrictivos que las interfaces petición-respuesta.

Control Jerárquico por Interfaces

Esta técnica utiliza *interfaces* que permite verificar las propiedades del sistema monolítico mediante un análisis local. Al evadir la necesidad de verificar las propiedades globalmente se logran ahorros computacionales y muchas veces se evita el problema de la explosión del espacio de estados.

El propósito de las interfaces es limitar la interacción entre los componentes de tal manera que las propiedades globales puedan ser verificadas localmente. Esta arquitectura es altamente reconfigurable ya que se puede modificar un componente del sistema sin tener que re-analizar el sistema global. Las restricciones sobre las interfaces puede resultar en un control sub-óptimo en el sentido de que se limitan las capacidades del sistema, como por ejemplo restringir al sistema a que realice dos tareas de manera secuencial mientras que la solución monolítica permite realizarlas simultáneamente sin violar especificaciones (en [16] se muestra un ejemplo de este caso); sin embargo en muchos casos es deseable la reducción de complejidad computacional y la reconfigurabilidad a cambio de la optimalidad.

El trabajo desarrollado sobre el uso de interfaces por Leduc, Brandin y Lawford [15] aborda el tema de las propiedades de controlabilidad y no bloqueo, pero en ese trabajo el manejo de interfaces se limita a dos niveles. Sin embargo, el trabajo de Lafortune, Tilbury, Queiroz, Cury y Hill [16] extiende las definiciones para el caso multi-nivel.

En la Figura 2.5 se muestra un ejemplo de sistema jerárquico multi-nivel con interfaces *petición-respuesta* donde cada componente de lazo cerrado H_k^i interacciona con los módulos vecinos mediante interfaces I_k^i.

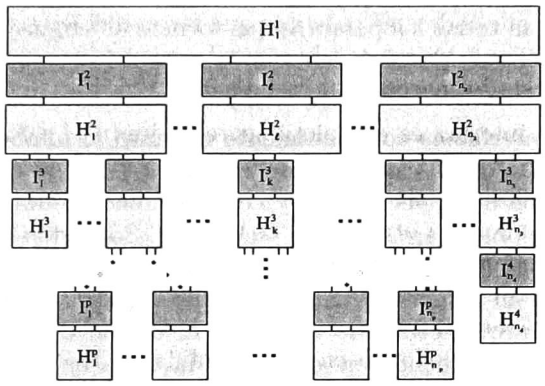


Figura 2.5: Arquitectura multinivel

Esta arquitectura generalizada ofrece ventajas significativas sobre la arquitectura de dos niveles propuesta por Leduc, ya que permite que el sistema sea particionado en módulos más pequeños, otra ventaja es la relajación de los requerimientos de consistencia en las interfaces.

Notación y requerimientos en los módulos

El sistema se encuentra dividido en módulos, cada uno consiste de una planta o subsistema G_k^i y un supervisor S_k^i construido con respecto a la especificación monolítica del nivel E_k^i resultando el subsistema de lazo cerrado $H_k^i = G_k^i \parallel S_k^i$. El superíndice i refleja el nivel de jerarquía y toma los valores $\{1, ..., p\}$. El subíndice k es para numerar los módulos e interfaces en cada nivel i y toma los valores $\{1, ..., n_i\}$.

Para cada subsistema en lazo cerrado H_k^i, el conjunto de índices de las interfaces I_k^i y de sus correspondientes módulos directamente debajo de ellas en la arquitectura de árbol serán identificados por la notación J_k^i. Al sistema global, en términos de estos módulos e interfaces, se le refiere como sistema Φ.

Todos los eventos compartidos entre un módulo H_k^i y sus vecinos del nivel superior se clasifican en eventos de petición $\rho \in \Sigma_{R_k^i}$ o eventos de respuesta $\alpha \in \Sigma_{A_k^i}$. La ocurrencia de cada uno de estos eventos debe ser aceptada por la interfaz correspondiente I_k^i. Conceptualmente, las peticiones son eventos que son controlados por el módulo de alto nivel y las respuestas son controladas por el módulo del nivel bajo.

Se asume que las interfaces de las que se habla tienen la forma *petición-respuesta.*

Definición 2.3.7 (Interfaz *petición-respuesta*). Un SED $I = (X, \Sigma_A \cup \Sigma_R, \xi, x_0, X_m)$ es una interfaz *petición-respuesta* si se cumple:

1. $L(I) \subseteq \overline{(\Sigma_R \Sigma_A)^*}$
2. $L_m(I) = (\Sigma_R \Sigma_A)^* \cap L(I)$

La primera significa que todas las palabras que forman el lenguaje de la interfaz deben tener la estructura petición-respuesta. Y la segunda quiere decir que toda respuesta a una petición resulta en un estado marcado de la interfaz.

El alfabeto de cada interfaz es completamente disjunto al alfabeto de cualquier otra interfaz. Además de que $\Sigma_{R_k^i} \cap \Sigma_{A_k^i} = \emptyset$. Todo módulo H_k^i puede compartir transiciones solamente con los módulos del nivel $(i+1)$ y con un único módulo del nivel $(i-1)$, lo que resulta en la ya mencionada arquitectura tipo árbol. Son estas restricciones sobre las interfaces y sobre la compartición de eventos en los módulos que hacen que pueda resultar en un control sub-óptimo en el sentido ya mencionado. Esto también dependerá en gran medida del diseño de la arquitectura de control y el diseño de las interfaces, es decir, la aplicación de esta técnica depende en gran medida de la experiencia del diseñador y del entendimiento del proceso: para particionar los componentes del sistema en módulos, seleccionar los eventos de petición-respuesta y construir las interfaces.

Se asume que el alfabeto global se particiona de la siguiente manera:

$$\Sigma := \Sigma_1^1 \dot{\cup} \left(\bigcup_{i=2,\ldots,p} \left(\bigcup_{i=2,\ldots,n_i} \left(\Sigma_k^i \dot{\cup} \Sigma_{A_k^i} \dot{\cup} \Sigma_{R_k^i} \right) \right) \right) \tag{2.1}$$

Donde el conjunto Σ_k^i representa a aquellos eventos que están en el conjunto H_k^i, pero en ningún conjunto de eventos de otro módulo; es decir, $\Sigma_k^i \cap \Sigma_{H_{k'}^{i'}} = \emptyset$, para toda $i \neq i'$ ó $k \neq k'$. Una consecuencia directa de la ecuación (2.1) es que cada interfaz es completamente disjunta de cualquier otra, es decir, $\Sigma_{I_k^i} \cap \Sigma_{I_{k'}^{i'}} = \emptyset$, para toda $i \neq i'$ ó $k \neq k'$ Además, el conjunto de peticiones $\Sigma_{R_k^i}$ es disjunto del conjunto de respuestas $\Sigma_{A_k^i}$. También se asume que el conjunto de eventos para cada módulo H_k^i se debe restringir de manera que se tenga la partición:

$$\Sigma_{H_k^i} = \Sigma_k^i \dot{\cup} \Sigma_{I_k^i} \cup \left(\bigcup_{j \in J_k^i} \Sigma_{I_j^{i+1}} \right) \tag{2.2}$$

La ecuación (2.2), es consistente con la mencionada estructura de árbol, significa que cada módulo H_k^i puede compartir transiciones solamente con los módulos del nivel $(i+1)$ y con un único módulo del nivel $(i-1)$.

Se empleará la siguiente notación para representar los lenguajes generados por los autómatas correspondientes, en términos del alfabeto global Σ.

- $P_{H_k^i} : \Sigma^* \to \Sigma_{H_k^i}^*$

  $\mathcal{H}_k^i := P_{H_k^i}^{-1}(L(H_k^i))$
- $\mathcal{G}_k^i := P_{H_k^i}^{-1}(L(G_k^i))$
- $\mathcal{S}_k^i := P_{H_k^i}^{-1}(L(S_k^i))$
- $\ell_k^i := P_{I_k^i}^{-1}(L(I_k^i))$

- $P_{I_k^i} : \Sigma^* \to \Sigma_{I_k^i}^*$
- $\mathcal{H}_{m_k}^i := P_{H_k^i}^{-1}(L_m(H_k^i))$
- $\mathcal{G}_{m_k}^i := P_{H_k^i}^{-1}(L_m(G_k^i))$
- $\mathcal{S}_{m_k}^i := P_{H_k^i}^{-1}(L_m(S_k^i))$

  $\ell_{m_k}^i := P_{I_k^i}^{-1}(L_m(I_k^i))$

Definición 2.3.8 (No bloqueo multinivel). Un sistema jerárquico multi-nivel Φ es no bloqueante multi-nivel si para todo $i \in \{1, ..., p\}$ y para toda $k \in \{1, ..., n_i\}$ para cada i, se satisface que:

$$\overline{\mathcal{H}_{m_k}^i \cap \ell_{m_k}^i \cap \bigcap_{j \in J_k^i} \ell_{mj}^{i+1}} = \mathcal{H}_k^i \cap \ell_k^i \bigcap_{j \in J_k^i} \ell_j^{i+1}$$

Definición 2.3.9 (Controlabilidad multi-nivel). El sistema jerárquico multi-nivel Φ es controlable multi-nivel con respecto a las particiones del alfabeto dadas por las ecuaciones (2.1) y (2.2), si para toda $i \in \{1, ..., p\}$ y toda $k \in \{1, ..., n_i\}$ para cada i, se satisface que:

I. El conjunto eventos de G_k^i y S_k^i es $\Sigma_{H_k^i}$ y el conjunto de eventos de I_k^i es $\Sigma_{J_k^i}$.

II. $\left(\forall s \in \mathcal{G}_k^i \cap \ell_k^i \cap \bigcap_{j \in J_k^i} \ell_j^{i+1} \cap \mathcal{S}_k^i\right) Elig_{\mathcal{G}_k^i \cap \bigcap_{j \in J_k^i} \ell_j^{i+1}}(s) \cap \Sigma_u \subseteq Elig_{\mathcal{S}_k^i \cap l_k^i}(S)$.

Definición 2.3.10 (Consistencia débil multi-nivel). El sistema jerárquico multi-nivel Φ será consistente débil con respecto a las particiones del alfabeto dadas por las ecuaciones (2.1) y (2.2), si para toda $s \in \mathcal{H}_k^i \cap \ell_k^i \bigcap_{j \in J_k^i} \ell_j^{i+1}$, para toda $i \in \{1, ..., p\}$ y para toda $k \in \{1, ..., n_i\}$ para cada i, las siguientes condiciones se satisfacen:

1. El conjunto evento de H_k^i es $\Sigma_{H_k^i}$.
2. I_k^i es una interfaz petición-respuesta, $i > 1$.
3. $Elig_{\ell_{j'}^{i+1}}(s) \cap \Sigma_{A_{j'}^{i+1}} \subseteq \Sigma_{\mathcal{H}_k^i}(s), \forall j' \in J_k^i$
4. $\left(\forall \rho \in \Sigma_{R_k^i}\right) s\rho \in \ell_k^i \Rightarrow \left(\exists l \in \Sigma_{L_k^i}^*\right) sl\rho \in \mathcal{H}_k^i \cap \ell_k^i \cap \bigcap_{j \in J_k^i} \ell_j^{i+1}$
5. $\left(\forall \alpha \in \Sigma_{A_k^i}\right) \left(s\rho \in \mathcal{H}_k^i \cap \ell_k^i \cap \bigcap_{j \in J_k^i} \ell_j^{i+1}\right) \wedge (s\rho\alpha \in \ell_k^i) \Rightarrow \left(\exists l \in \Sigma_{L_k^i}^*\right) s\rho l\alpha \in \mathcal{H}_k^i \cap \ell_k^i \cap \bigcap_{j \in J_k^i} \ell_j^{i+1}$
6. $s \in \ell_{m_k^i} \Rightarrow \left(\exists l \in \Sigma_{L_k^i}^*\right) sl \in \mathcal{H}_{m_k}^i \cap \ell_{m_k}^i \cap \bigcap_{j \in J_k^i} \ell_{m_j}^{i+1}, i > 1$

El punto 3 de la definición significa que el lenguaje de cada módulo $\mathcal{H}_k^i$ sea $\Sigma_{A_k^i}$ $-controlable$ con respecto a cada una de las interfaces del nivel inmediato inferior. Los puntos 4 y 5 quieren decir que los eventos de petición y de respuesta, sean alcanzables en el módulo mediante eventos no compartidos con el módulo del nivel alto que corresponde. El punto 6 significa que si una palabra es parte del lenguaje marcado de la interfaz, entonces esta palabra puede ser extendida a una que sea parte del lenguaje de aceptación en el nivel bajo correspondiente, mediante eventos que no sean compartidos con el módulo del nivel alto.

Teorema 2.3.5. Si el sistemas de dos niveles basado en interfaces compuesto por los módulos $H^1, H^2, I_1^2, ..., H_n^2, I_n^2$, es no bloqueante y consistente débil con respecto a la partición del alfabeto dada por (1), entonces el sistema global es no bloqueante:

$$\overline{\mathcal{H}_m^1 \cap \bigcap_{j=1,...,n} \left(\mathcal{H}_{m_j}^2 \cap \ell_{m_j}^2\right)} = \mathcal{H}^1 \cap \bigcap_{j=1,...,n} \left(\mathcal{H}_j^2 \cap \ell_j^2\right).$$

Teorema 2.3.6. Si el sistemas de dos niveles basado en interfaces compuesto por los subsistemas $G^1, G^2, ..., G^n$, supervisores $S^1, S^2, ..., S_n^2$, e interfaces $I_1^2, ..., I_n^2$, es controlable con respecto a la partición del alfabeto dado por (2.1), entonces el lenguaje del supervisor $\mathcal{S} = \mathcal{S}^1 \cap \bigcap_{j=1,...,n} \left(\mathcal{S}_j^2 \cap \ell_j^2\right)$ es Σ_u $-$ *controlable* con respecto al lenguaje de la planta $\mathcal{G} = \mathcal{G}^1 \cap \bigcap_{j=1,...,n} \mathcal{G}_j^2$.

Teorema 2.3.7. Si el sistema jerárquico multi-nivel Φ es multi-nivel no bloqueante y consistente débil respecto a las particiones del alfabeto dado por (2.1) y (2.2), entonces el sistema global es no-bloqueante:

$$\overline{\mathcal{H}_m^1 \cap \mathcal{H}_m^2 \cap \ell_m^2 \cap \cdots \mathcal{H}_m^p \cap \ell_m^p} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \ell^2 \cap \cdots \mathcal{H}^p \cap \ell^p$$

Donde:

$$\begin{aligned} \mathcal{H}_m^i &= \mathcal{H}_m^i \cap \cdots \cap \mathcal{H}_{m_{n_i}}^i,\ \ell_m^i = \ell_m^i \cap \ldots \ell_{m_{n_i}}^i, \\ \mathcal{H}^i &= \mathcal{H}_1^i \cap \cdots \cap \mathcal{H}_{n_i}^i,\ \ell^i = \ell_1^i \cap \ldots \ell_{n_i}^i \end{aligned}$$

Teorema 2.3.8. Si el sistema jerárquico multi-nivel Φ es controlable multi-nivel con respecto a las particiones del alfabeto dadas por (2.1) y (2.2), entonces el lenguaje supervisor $\mathcal{S} = \mathcal{S}^1 \cap \mathcal{S}^2 \cap \ell^2 \cap \cdots \cap \mathcal{S}^p \cap \ell^p$ es Σ_u $-$ *controlable* con respecto al lenguaje de la planta $\mathcal{G} = \mathcal{G}^1 \cap ... \cap \mathcal{G}^p$, donde $\mathcal{S} = \mathcal{S}_1^i \cap ... \cap \mathcal{S}_{n_i}^i$, $\ell^i = \ell_1^i \cap ... \cap \ell_{n_i}^i$, y $\mathcal{G}^i = \mathcal{G}_1^i \cap ... \cap \mathcal{G}_{n_i}^i$.

Consultar [15] y [16] para ver las pruebas de estos teoremas.

2.4. Controladores Lógicos Programables

Los Controladores Lógicos Programables, también llamados autómatas programables o PLC (por sus siglas en inglés), son dispositivos de estado sólido de la familia de

computadoras que emplean circuitos integrados en vez de dispositivos electromecánicos para implementar funciones de control. Son capaces de almacenar instrucciones, tales como la secuenciación, el tiempo, operaciones aritméticas, contadores, manipulación de datos y comunicación; para controlar máquinas y procesos industriales.

2.4.1. La norma IEC 1131

En la actualidad aún siguen persistiendo sistemas de control específicos del fabricante, con programación dependiente y conexión compleja entre distintos sistemas de control. Esto significa para el usuario costos elevados, escasa flexibilidad y falta de normalización en las soluciones al control industrial. la norma IEC 1131 es el primer paso en la estandarización de los autómatas programables y sus periféricos, incluyendo los lenguajes de programación que se deben utilizar. Enseguida se presenta una traducción libre, comentada y resumida de [36] y [37]. Se puede dividir el estándar en dos partes:

- Elementos comunes.
- Lenguajes de programación.

Elementos comunes

Tipos de datos

Los tipos comunes de datos son: variables Booleanas, número entero, número real, byte y palabra, pero también fechas, horas del día y cadenas (strings). El usuario puede definir los propios, conocidos como tipos de datos derivados.

Variables

Las variables permiten identificar los objetos de datos cuyos contenidos pueden cambiar, por ejemplo, los datos asociados a entradas, salidas o a la memoria del autómata programable. Una variable se puede declarar como uno de los tipos de datos elementales definidos o como uno de los tipos de datos derivados. De este modo se crea un alto nivel de independencia con el hardware, favoreciendo la reusabilidad del software. La extensión de las variables está normalmente limitada a la unidad de organización en la cual han sido declaradas como locales. Esto significa que sus nombres pueden ser reutilizados en otras partes sin conflictos, eliminando una frecuente fuente de errores. Si las variables deben tener una extensión global, han de ser declaradas como globales.

Configuración, recursos y tareas

Al más alto nivel, el elemento de software requerido para solucionar un problema de control particular puede ser formulado como una configuración. Una configuración es específica para un tipo de sistema de control, incluyendo las características del hardware: procesadores, direccionamiento de la memoria para los canales de I/O y otras capacidades del sistema.

Unidades de Organización de Programa

Dentro de IEC 1131-3, los programas, bloques funcionales y funciones se denominan Unidades de Organización de Programas (POU's por sus siglas en inglés).

Funciones

IEC 1131-3 especifica funciones estándar y funciones definidas por usuario. Las funciones estándar son por ejemplo ADD (suma), ABS (valor absoluto), SQRT (raíz cuadrada), SIN (seno), y COS (coseno). Las funciones definidas por usuario, una vez implementadas pueden ser usadas indefinidamente en cualquier POU. Las funciones no pueden contener ninguna información de estado interno, es decir, que la invocación de una función con los mismos argumentos (parámetros de entrada) debe suministrar siempre el mismo valor (salida).

Bloques Funcionales (FB's por sus siglas en inglés)

Los bloques funcionales son los equivalentes de los circuitos integrados, IC's por sus siglas en inglés, que representan funciones de control especializadas. Los FB's contienen tanto datos como instrucciones, y además pueden guardar los valores de las variables (que es una de las diferencias con las funciones). Tienen un interfaz de entradas y salidas bien definido y un código interno oculto, como un circuito integrado o una caja negra. De este modo, establecen una clara separación entre los diferentes niveles de programadores, o el personal de mantenimiento. Un lazo de control de temperatura, PID, es un ejemplo de bloque funcional. Una vez definido, puede ser usado una y otra vez, en el mismo programa, en diferentes programas o en distintos proyectos. Esto lo hace altamente reutilizable. Los bloques funcionales pueden ser escritos por el usuario en alguno de los lenguajes de la norma IEC, pero también existen FB's estándar (biestables, detección de flancos, contadores, temporizadores, etc.). Existe la posibilidad de ser llamados múltiples veces creando copias del bloque funcional que se denominan instancias. Cada instancia llevará asociado un identificador y una estructura de datos que contenga sus variables de salida e internas.

Programas

Los programas son un conjunto lógico de todos los elementos y construcciones del lenguaje de programación que son necesarios para el tratamiento de señal previsto que se requiere para el control de una máquina o proceso mediante el sistema de autómata programable. Un programa puede contener, aparte de la declaración de tipos de datos, variables y su código interno, distintas instancias de funciones y bloques funcionales.

Lenguajes de Programación

Se definen cuatro lenguajes de programación normalizados, la Figura 2.6 muestra un ejemplo en donde se describe la misma acción con los cuatro lenguajes. Esto significa que su sintaxis y semántica ha sido definida, no permitiendo particularidades distintivas (dialectos). Una vez aprendidos se podrá usar una amplia variedad de sistemas basados en esta norma. Los lenguajes consisten en dos de tipo literal y dos de tipo gráfico:

- Literales: Lista de instrucciones (Instruction List, IL) y Texto estructurado (Structured Text, ST).
- Gráficos: Diagrama de escalera (Ladder Diagram, LD) y Diagrama de bloques funcionales (Function Block Diagram, FBD).

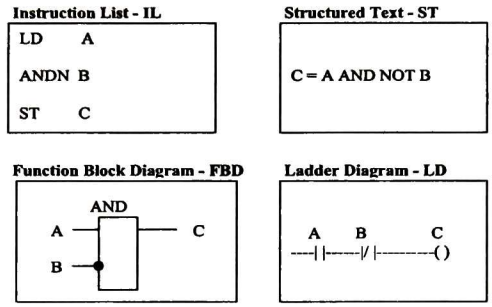


Figura 2.6: Lenguajes IEC 1131-3

Los cuatro lenguajes pueden ser empleados para resolver conjuntamente un problema común según la experiencia del usuario.

El Diagrama de escalera (LD) tiene sus orígenes en los Estados Unidos. Está basado en la presentación gráfica de la lógica de relés. Lista de Instrucciones (IL) es el modelo de lenguaje ensamblador basado en un acumulador simple; procede del alemán Anweisungliste, AWL. El Diagramas de Bloques Funcionales (FBD) es muy común en aplicaciones que implican flujo de información o datos entre componentes de control. Las funciones y bloques funcionales aparecen como circuitos integrados y es ampliamente utilizado en Europa. El lenguaje Texto estructurado (ST) es un lenguaje de alto nivel con orígenes en el Ada, Pascal y 'C'; puede ser utilizado para codificar expresiones complejas e instrucciones anidadas; este lenguaje dispone de estructuras para bucles (REPEAT-UNTIL; WHILE-DO), ejecución condicional (IF-THEN-ELSE; CASE), funciones (SQRT, SIN, etc.).

Muchos entornos de programación IEC actuales ofrecen aquello que se espera a nivel de interface de usuario: uso de ratón, menús desplegables, pantallas de programación gráfica, múltiples ventanas, ayuda en línea, verificación durante el diseño, etc. Debe hacerse notar que estos detalles no están especificados en la norma por lo que es una de las partes donde los proveedores pueden diferenciarse.

Diagrama de Escalera (LD)

La evolución del lenguaje de escalera original ha ido añadiendo instrucciones cada vez más poderosas. Estas nuevas funciones han sido agregadas a las operaciones básicas de relé, temporizado y conteo. El término función es utilizado para describir instrucciones, que como su nombre lo implica, llevan a cabo funciones de transferencia de datos dentro del PLC. Estas instrucciones se basan en los principios de la lógica básica de relés, aunque permiten la implementación de operaciones complejas. Las adiciones a la lógica básica

de relés incluyen bloques de función, que usan un conjunto de instrucciones para operar sobre un bloque de información. El uso de los bloques de función incrementa la potencia del lenguaje de escalera básico, conformando lo que se conoce como *lenguaje de escalera mejorado.*

De esta manera, el lenguaje de escalera disponible para la programación del PLC se puede dividir en dos grupos:

- Lenguaje de escalera básico
- Lenguaje de escalera mejorado

La línea que define la agrupación de instrucciones es típicamente establecida entre categorías funcionales. Estas categorías incluyen:

- Relé
- Temporizado
- Conteo
- Control de programa y flujo
- Aritmética
- Manipulación de datos
- Transferencia de datos
- Funciones especiales (secuenciadores)
- Comunicación a través de la red

Las funciones principales de un programa en LD están orientadas a controlar salidas y ejecutar funciones operacionales, basado en la condición de entradas. Se utilizan peldaños o segmentos (*rung* en inglés) para acometer su control. En general, un peldaño consiste de un conjunto de condiciones de entrada (representado por instrucciones de contactos) y una instrucción de salida al final del peldaño (representado por un símbolo de bobina). Estos elementos de contacto se cierran o abren de acuerdo al estado de las entradas referenciadas.

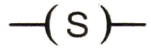
Enseguida se detallarán algunas de las categorías en que se agrupan las instrucciones de acuerdo a su función, solamente se explicarán aquellas empleadas en este trabajo, para más información consultar [40].

Instrucciones de Relés

Las instrucciones de relé son las instrucciones más básicas de LD y usan dos tipos de símbolos: *contactos* y *bobinas.* Estas instrucciones representan estados encendido/apagado de las entradas y salidas conectadas. Los contactos simbolizan las condiciones de entrada que deben ser evaluadas en un peldaño específico, para determinar el control de la salida. Las bobinas corresponden a las salidas de los peldaños.

En un programa, cada contacto y bobina está referenciado con una dirección, la cual identifica aquello que está siendo evaluado o controlado. La dirección hace referencia a la entrada o salida conectada al PLC o al bit de almacenamiento interno (estado interno o relé de control). En la Tabla 2.1 se muestran las instrucciones de relé empleadas en este trabajo.

Tabla 2.1: Instrucciones de relé empleadas

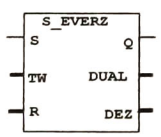
Instrucciones de Relé		
Instrucción	Símbolo	Función
Contacto Normalmente Abierto		Contacto que se cierra cuando la entrada referenciada es verdadera (hay un uno lógico en el elemento de entrada asociado).
Bobina regular o de salida		Mantiene su salida en alto mientras su entrada esté en alto.
Bobina de enganche		Mantiene la salida en alto una vez que el elemento ha sido energizado.
Bobina de liberación		Desactiva (coloca en cero lógico) la salida previamente activada por la bobina de enganche.

Instrucciones de Temporización y Conteo

Los temporizadores y contadores son instrucciones internas que proveen las mismas funciones que sus contrapartes de hardware. Activan o desactivan un dispositivo luego de que transcurre un determinado lapso de tiempo o de que se ha alcanzado un conteo específico. Los temporizadores y contadores son típicamente considerados como salidas internas.

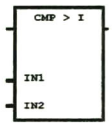
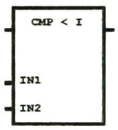
Las instrucciones de temporizador pueden tener una o más bases de tiempo que son usadas para cronometrar un evento. Los temporizadores son usados para añadir un determinado lapso de tiempo a una salida del programa. La Tabla 2.2 muestra el único elemento de temporización empleado en la tesis.

Tabla 2.2: Instrucciones de temporización empleadas

Instrucciones de Temporizado		
Instrucción	Símbolo	Función
Temporizador		Cuando la entrada S está en alto, el temporizador inicia su cuenta (el tiempo en ms se coloca en la entrada TW). Cuando éste termina su cuenta se activa la salida Q.

Instrucciones de Comparación de Datos
Las instrucciones de comparación de datos como lo implica su nombre, comparan valores almacenados en dos registros. Estas instrucciones son útiles cuando se verifica el rango de datos de una variable. En algunos PLC's, las instrucciones de comparación de datos son expresadas en formato básico de escalera, mientras que en otros, son bloques funcionales. En ambos formatos, proporcionan tres comparaciones básicas: igual a, menor que y mayor que. Con base en el resultado de la comparación, el procesador puede encender o apagar salidas y llevar a cabo otras operaciones. La Tabla 2.3 muestra los comparadores empleados en este documento.

Tabla 2.3: Instrucciones de comparación empleadas

Instrucciones de Comparación de Datos		
Instrucción	Símbolo	Función
Comparador >		Coloca su salida en alto cuando IN1 es mayor a IN2
Comparador <		Coloca su salida en alto cuando IN1 es menor a IN2

2.4.2. Ventajas del uso de PLC's

En general, la arquitectura de un PLC es modular y flexible, permitiendo expandir con facilidad elementos de software y hardware conforme los requerimientos de la aplicación cambian. En caso de que la aplicación sobrepase las limitaciones del PLC, la unidad puede ser reemplazada fácilmente por otra con mayor memoria o capacidad de entradas y salidas. Un sistema que emplea un PLC provee de muchos beneficios en cuanto a soluciones de control, desde confiabilidad y repetibilidad hasta programabilidad, permitiendo realizar modificaciones sin cambiar el cableado. Entre otras ventajas está su costo accesible y mantenimiento económico.

A continuación se enlistan las ventajas y desventajas que trae consigo el empleo de un PLC.
Ventajas:

- Control más preciso.
- Mayor rapidez de respuesta.
- Flexibilidad en el control de procesos complejos.
- Facilidad de programación.
- Seguridad en el proceso.

- Empleo de poco espacio.
- Fácil instalación.
- Menos consumo de energía.
- Mejor monitoreo del funcionamiento.
- Menor mantenimiento.

  Detección rápida de averías y tiempos muertos.

  Menor tiempo en la elaboración de proyectos.
- Posibilidad de añadir modificaciones sin elevar costos.
- Menor costo de instalación, operación y mantenimiento.
- Posibilidad de gobernar varios actuadores con el mismo autómata.

Desventajas:

- Mano de obra especializada.
- Centraliza el proceso.
- Condiciones ambientales apropiadas.
- Mayor costo cuando se trata de controlar tareas sencillas.

Capítulo 3

Implementación Formal de Sistemas de Control Jerárquico-Modular

En este capítulo se muestra una forma general de implantar la capa I/O entre el sistema a controlar y los supervisores (capa de comunicación), se expone una manera de representar las máquinas de estados finitos que modelan a los componentes elementales del sistema en lenguaje de escalera. Posteriormente en la sección 3 se presenta una traducción a código implementable de los supervisores representados como autómatas, esta traducción es diferente dependiendo de la técnica de control jerárquico empleada. Para el caso de la técnica de *Control por Eventos Imperativos* se hace una traducción a lenguaje estructurado y para *Vocalización* se establece una traducción formal a diagrama de escalera. Se hace uso de la terminología definida por la norma ISA-S88 para referirse a los diferentes niveles de supervisión, sin embargo la implementación no está enfocada únicamente a sistemas de manufactura.

3.1. Representación de MEF que modelan a los componentes elementales en LD

Los modelos en MEF que corresponden a los componentes elementales del sistema se implementan empleando una traducción ad-hoc entre MEF y diagrama de escalera (LD). Esta implementación corresponde a la capa de comunicación o capa I/O entre los supervisores y el sistema a controlar. Los componentes en LD empleados para representar la estructura G son solamente bobinas de enganche (*latch coils* o *set coils* en inglés), bobinas de liberación (*unlatch coils* o *reset coils* en inglés), bobinas regulares y contactos normalmente abiertos.

Los **eventos incontrolables** están relacionados con salidas del sistema que pueden ser analógicas o digitales correspondientes a sensores, temporizadores, contadores, entre otros. Estas salidas de la planta son percibidas por el sistema de control para tomar decisiones. De esta manera, en la implementación de los eventos incontrolables en la capa de comunicación sistema-control es preciso emplear diversos **bloques de función** puesto

que la recepción y entrega de datos depende de la naturaleza de lo que se está sensando. Se emplearán **comparadores** en aquellos casos donde se tiene un sensor analógico y se quiere medir un rango (por ejemplo que los datos medidos no pasen de cierto valor o que estén en un rango determinado). Se utilizarán **temporizadores** para todas las situaciones en que se requiera medir el tiempo. Se usarán **bobinas regulares** para los casos en que se esté sensando una entrada digital (por ejemplo monitorear si se presiona un botón).

La estructura formal de una MEF representada con LD está dada por:

$$L := (C, \Pi, \gamma, c_0, 2^S, 2^R, 2^B)$$

Donde:

- C y Π son conjuntos de contactos normalmente abiertos con variables asociadas de tipo booleano, para modelar los estados (contacto de estado) y a las transiciones (contacto de transición) respectivamente. Existen los mapeos $\Psi : Q \to C$ y $\Phi : \Sigma \to \Pi$ que relacionan a éstos conjuntos de contactos normalmente abiertos con el conjunto de estados y transiciones. Los mapeos inversos $\Psi^{-1} : C \to Q$ y $\Phi^{-1} : \Pi \to \Sigma$ también están definidos. La etiqueta del contacto de estado $c \in C$ será "Nombre_Sxx" donde "Nombre" es la abreviación del componente elemental correspondiente y "xx" es el número del estado de la MEF que se está modelando.
- $\gamma : C \times \Pi \to 2^S \times 2^R \times 2^B$ es la función parcial de transición que refleja la función parcial δ.
- c_0 es el estado inicial.
- 2^S, 2^R y 2^B son conjuntos de conjuntos de bobinas de enganche, bobinas de liberación y bobinas regulares respectivamente, empleadas para cerrar o abrir los contactos normalmente abiertos que modelan los estados y las transiciones (modificando el estado de las variables booleanas asociadas a ellos) y para notificar los eventos incontrolables.

El morfismo de δ a γ se construye como sigue: dada una MEF con $q \in Q$ y $\sigma \in \Sigma$, si $\delta(q, \sigma)$ está definida, entonces $\gamma(\Psi(q), \Phi(\sigma))$ debe también estar definida, sin embargo hay una diferencia sutil en la implementación de la función γ dependiendo si $\sigma \in \Sigma_c$ ó $\sigma \in \Sigma_u$ por lo que se distinguen dos casos en la traducción.

3.1.1. Implementación en LD cuando $\sigma \in \Sigma_c$

La función $\gamma(\Psi(q), \Phi(\sigma))$ está definida de la siguiente manera:

$$\gamma(c, \pi) = (\{s, S'\}, \{r, R'\, r''\}, \{\})$$

con $\tilde{c} = \Psi(\delta(q, \sigma))$. Donde s representa la bobina de enganche asociada al contacto de estado $\tilde{c} \in C$ (i.e el estado siguiente en la MEF). S' y R' denotan conjuntos de bobinas

de enganche y liberación relacionadas con uno o más pines de salida del PLC que se activan/desactivan por la ejecución de la transición σ (i.e. $\Phi(\sigma) = \pi$).

En sistemas no muy complejos, por lo general sólo habrá un elemento en el conjunto S' y ninguno en el conjunto R', es decir a cada transición corresponde la activación de un solo pin de salida.

r representa la bobina de liberación asociada al contacto de estado $c \in C$ (i.e estado presente en la MEF) y r'' es la bobina de liberación asociada con el contacto de transición π que acaba de ocurrir.

De manera intuitiva, $\gamma(c, \pi) = (\{s, S'\}, \{r, R'\, r''\}, \{\})$ se representa como el contacto $c \in C$ en serie con uno o más contactos $\pi \in \Pi$ en paralelo (dependiendo si en la MEF sale una o más transiciones del estado actual) seguido por el conjunto de bobinas de enganche y liberación en paralelo. Es decir, una de las bobinas de enganche es utilizada para poner en alto el valor del contacto del siguiente estado $\tilde{c} \in C$ mientras que la segunda se emplea para colocar en alto la salida correspondiente al evento ocurrido). De manera correspondiente, una de las bobinas de liberación es utilizada para poner en bajo el valor del contacto del estado presente mientras que la segunda pone en bajo el contacto correspondiente al evento controlable que recién ocurrió.

3.1.2. Implementación en LD cuando $\sigma \in \Sigma_u$

En este caso la función $\gamma(\Psi(q), \Phi(\sigma))$ está definida de la siguiente manera: $\gamma(c, \pi) = (\{s, s'\}, \{r\}, \{b\})$ con $\tilde{c} = \Psi(\delta(q, \sigma))$. En este caso no se requiere la bobina de liberación r' como en el caso cuando $\sigma \in \Sigma_c$ puesto que la ocurrencia del evento incontrolable $\sigma \in \Sigma_u$ se hace presente mediante la bobina regular b, la cual mantiene su salida en alto mientras la entrada esté en alto (i.e. mientras el evento incontrolable esté presente, la variable de transición correspondiente está activa y se desactiva automáticamente sin necesidad de una bobina de liberación).

3.2. Implementación: Control por Eventos Imperativos

3.2.1. Arquitectura de supervisión y comunicación entre niveles

Los supervisores se asocian a las *fases* de equipo y *procedimientos unitarios*, los cuales se encargan de hacer cumplir las especificaciones de seguridad, procedimiento, compartición de recursos, etc. Por otro lado el *procedimiento de receta* se encarga únicamente de establecer la secuencia deseada de fases para manufacturar un producto específico.

En esta técnica, los supervisores sintetizados con TCS son implementados como dispositivos de control. Es decir, bajo ciertas circunstancias el supervisor es capaz de *ejecutar* transiciones controlables. Cuando en un estado solamente hay una transición controlable, entonces el supervisor tiene la capacidad de ejecutarla. Si se tienen varias incontrolables y una controlable, entonces se da prioridad a la presencia de las transiciones incontrolables (ya que el código es secuencial y se tiene que definir una prioridad) y en caso de que no se haya presentado ningún evento incontrolable, se ejecuta la transición controlable. Cuando de un estado salen dos o más transiciones controlables, éstas deben ser declaradas como *imperativas*; las cuales, como se expondrá en el siguiente punto, se implementan como entradas en todos los niveles de supervisión y son salidas en el nivel de *procedimiento de receta*. Es decir, los supervisores consideran todas las posibilidades en cuanto al orden en que se pueden realizar las tareas y es hasta el nivel de la *receta* donde se fija el orden exacto en que se deben ejecutar las tareas o procesos.

En el nivel más bajo de la arquitectura (nivel *fase*) se sintetizan supervisores modulares relacionados con el equipo, cada supervisor de *fase* contempla todas las posibilidades que el equipo que supervisa es capaz de realizar sin violar las especificaciones. Una vez sintetizados los supervisores de *fase* se procede a identificar las transiciones de inicio de tarea y declararlas como imperativas.

Para la síntesis del supervisor de un nivel superior (*operación* o *procedimiento unitario*) se realizan proyecciones naturales sobre los eventos imperativos e informativos de interés, tomando en cuenta los módulos de *fase* involucrados; resultando una serie de autómatas que modelan a un nivel más abstracto el comportamiento de las *fases*, éstos serán los componentes elementales del nivel superior, los cuales se sincronizan para obtener el modelo abstracto. Se diseñan especificaciones de seguridad o administración de recursos para este nivel superior y se obtiene el supervisor con el procedimiento estándar, explicado en la sección 2.1.2. Se extiende este mismo procedimiento hasta tener la cantidad requerida de capas de supervisión, por lo general para sistemas de complejidad media se tendrán dos niveles de supervisión y el nivel de la *receta*.

Por lo general, las palabras que forman el lenguaje de la *receta* tendrán la estructura *inicio de tarea fin de tarea*, la *receta* habilita las transiciones imperativas de los niveles inferiores y espera como entrada, de manera incontrolable, que se presente el fin de tarea; sin embargo no está limitado a que sea de esa manera, puede haber dos o más comandos imperativos consecutivos y después de ellos los informativos correspondientes. Es decir, el lenguaje de aceptación del autómata que describe a la *receta* está dado de la siguiente manera:

$$L(M) \subseteq \left\{ \Sigma_{imp}^{n} \Sigma_{inf}^{n} \mid n \in \mathbb{N} \right\}$$

La comunicación entre las capas de supervisión se da de la siguiente manera: la *receta* da la orden de inicio de tarea (ejecución de la transición imperativa de habilitación), el

nivel inferior recibe dicha orden como entrada y comienza a realizar el proceso, al finalizar notifica (i.e. ejecución de una transición controlable) al supervisor de nivel superior quien espera esa notificación como una transición incontrolable. Proponer esta forma de comunicación implica que la naturaleza de algunas transiciones (aquellas utilizadas como mecanismo de sincronización entre supervisores) cambie según el nivel de supervisión, sin embargo simplifica su implementación al no tener que modelar por separado los canales de comunicación.

Es suficiente implementar solamente las *recetas* (puede haber una *receta* por cada módulo o una sola *receta*, dependiendo el modelo de control de procedimientos que se haya diseñado) y los supervisores de *fase*; los niveles intermedios únicamente servirán para verificar la realizabilidad de la *receta* de manera previa a la implementación. No representa un problema implementar todos los niveles de supervisión, sin embargo resultaría un desperdicio de recursos que no aporta beneficios notables. Basta con hacer la prueba de realizabilidad para garantizar que el sistema en lazo cerrado tendrá un comportamiento adecuado y se logrará la manufactura del producto indicado por la *receta*.

3.2.2. Modificaciones en la teoría

Inicialmente se había definido que el lenguaje del *procedimiento de receta* estaba construido únicamente por transiciones imperativas, sin embargo no había un mecanismo de sincronización explícito entre la *receta* y los niveles inferiores. Es decir, la *receta* ejecutaba la transición imperativa y no había un elemento que indicara el lapso de espera hasta la ejecución de la siguiente transición imperativa, sino que podría malinterpretarse como una ejecución serial de todas las imperativas, sin sincronía con los niveles inferiores. De esta manera, se modificó el lenguaje del *procedimiento de receta* de tal manera que después de la ejecución de la transición imperativa se encuentre una transición incontrolable a este nivel que será una entrada para esperar a que el supervisor inferior le indique la finalización de la tarea y pueda proceder con la siguiente imperativa. De esta manera, son necesarias solamente dos modificaciones en las definiciones presentadas en la sección 2.3.1 :

- $M := (Q, \Sigma, \delta, q_o, Q_m)$ es una MEF cuyo lenguaje describe el conjunto de secuencias de comandos de fabricación para un producto en términos de eventos imperativos y sus informativos correspondientes.
- La definición de la proyección que aparece en la definición de lenguaje realizable y la proposición 2.2.1 se modifica para que el resto de la teoría se preserve:

$$Proj : \Sigma^* \rightarrow \Sigma_{L(M)}$$

El cambio de naturaleza que presentan algunas transiciones informativas (de ser controlables en el nivel *fase* pasan a ser incontrolables en el nivel *operación*) no afecta las definiciones de confiabilidad ni realizabilidad. En la definición de autómata

confiable solamente se toman en cuenta las transiciones imperativas, las cuales son controlables en todos los niveles. Para la realizabilidad de la receta basta con probar que $Proj\,(L\,(M)\parallel L\,(G))$ sea isomorfo a $L\,(M)$, para lo cual tampoco afecta el cambio de naturaleza de las transiciones informativas de fin de tarea.

3.2.3. Representación en Texto Estructurado (ST)

El texto estructurado es uno de los lenguajes apoyados por el estándar IEC 61131-3 diseñado para PLC. Es un lenguaje de alto nivel estructurado por bloques y que se asemeja sintácticamente a Pascal. Cada supervisor obtenido mediante la técnica de control por eventos imperativos se puede representar como una función programada en ST.

Un supervisor $S = (Y, \Sigma, \delta_S, y_0, Y_m)$ y $\phi : Y \rightarrow \Gamma$ se traduce a ST de la siguiente manera; el conjunto de estados Y es modelado por la variable *state* de tipo entero que toma los valores $0...(k-1)$ y de cardinalidad $k = |Y|$. El estado inicial y_0 se asocia con el valor 0 de la variable *state*. El resto de los valores de la variable *state* se asigna de acuerdo con la función parcial de transición δ_S. Los estados marcados Y_m no se requieren en la implementación en ST. Las transiciones controlables e incontrolables se modelan, respectivamente, por variables booleanas de salida (i.e. acciones a ser ejecutadas desde un estado dado) y variables booleanas de entrada (i.e. señales de entrada esperadas por el supervisor en un estado particular). Las transiciones imperativas se modelan mediante variables booleanas, las cuales serán de entrada en los niveles de supervisión y salidas en la *receta*.

La función de transición parcial $\delta_S : Y \times \Sigma \rightarrow Y$ es implementada mediante la estructura *case*, el programa interpreta el valor de la variable *state* almacenándolo en una zona de memoria temporal (oculta), después lo compara con el valor seguido de cada *case*, y si coincide, se realiza el flujo de acciones correspondientes (estas acciones son los eventos que ocurren en el estado en cuestión), diferenciando la naturaleza de las transiciones habilitadas y priorizando la presencia de transiciones incontrolables.

Si una variable de entrada es verdadera (i.e. se presenta una transición incontrolable esperada en ese estado o se encuentra habilitada una transición imperativa), se ejerce un cambio en *state* (e.g. IF T312 = TRUE THEN STATE:=8 END_IF). Únicamente para el caso de que sea una transición imperativa, es necesario regresar al valor de falso la variable asociada.

Si existe una transición controlable habilitada en el estado, su variable booleana correspondiente se coloca en verdadero y a la variable *state* se le asigna el valor establecido por δ_S. Las variables de entrada y salida que corresponden a los eventos controlables e incontrolables son reseteadas (i.e. puestas en falso) en la implementación que corresponde a los componentes elementales hecha con lógica de escalera, a excepción de las transiciones controlables imperativas las cuales deben resetearse en la estructura IF-THEN en el bloque de sentencias cuando la condición es verdadera.

Ejemplos de traducción

En Figura 3.1 se muestra un ejemplo de traducción de MEF a ST para un estado en el que están habilitadas dos transiciones imperativas, éstas se implementan como entradas y serán comandos de salida solamente en el nivel *receta*.

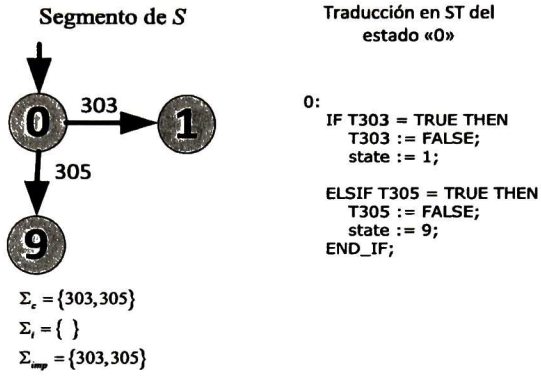


Figura 3.1: Ejemplo de traducción para eventos imperativos

En la Figura 3.2 se muestra la traducción de un estado en el que están habilitadas dos transiciones incontrolables y una controlable, se observa que se les da prioridad a la presencia de alguna incontrolable antes de la ejecución de la controlable.

En las Figuras 3.3 y 3.4 se muestra la implementación para el caso de un estado con una sola transición controlable y el caso para una incontrolable habilitada respectivamente.

En la Figura 3.5 se observa un ejemplo de implementación de un supervisor en texto estructurado como un bloque de función. La lógica empleada para la representación de supervisores en texto estructurado puede ser utilizada para implementarlos en diagrama de escalera. En la siguiente sección, los supervisores obtenidos por medio de *vocalización* se presenta su traducción a LD para ejemplificar el uso de los dos lenguajes más utilizados en la programación de PLC. Al presentar los esquemas de traducción de supervisores en dos lenguajes, mostrando los elementos empleados de cada lenguaje, es fácil entrever su implementación en cualquiera de los dos.

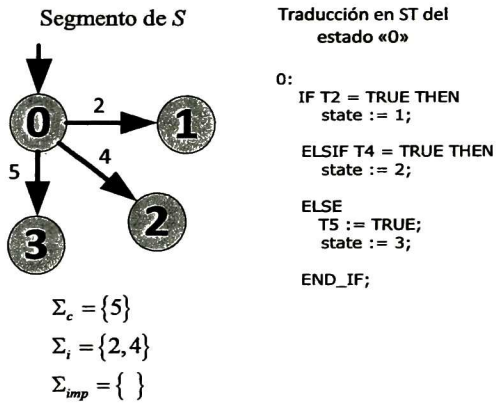


Figura 3.2: Ejemplo de traducción cuando en un estado se encuentran habilitadas transiciones incontrolables y una controlable

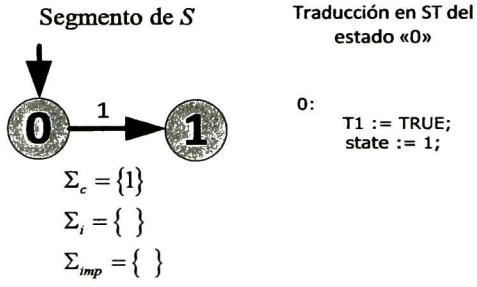


Figura 3.3: Traducción de eventos controlables

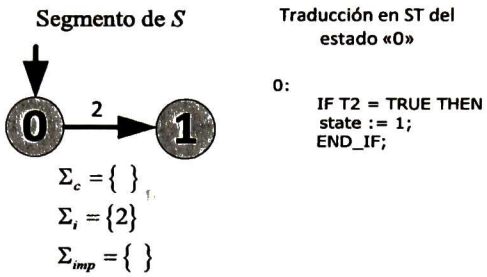


Figura 3.4: Traducción de eventos incontrolables

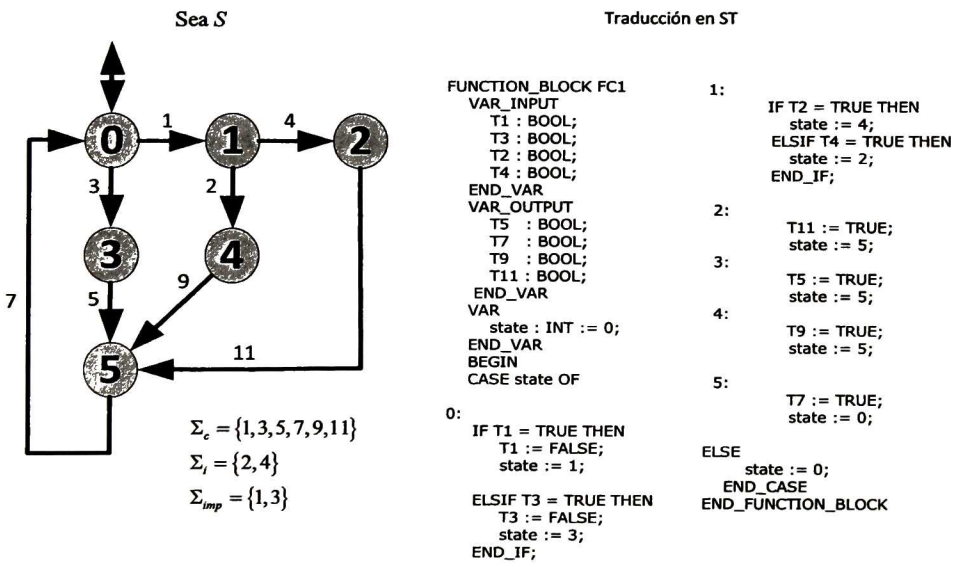


Figura 3.5: Ejemplo de Implementación de un Supervisor en ST

3.3. Implementación: Vocalización

3.3.1. Arquitectura de supervisión y comunicación entre niveles

Cada supervisor corresponde a un bloque de función, sin embargo, las MEF que representan los supervisores no muestran toda la información que está implícita en el esquema de comunicación entre supervisores, por lo cual es preciso disponer de más información obtenida en la etapa de diseño para poder lograr la traducción. De manera adicional a estos supervisores se implementó una función encargada de la inicialización de todos los supervisores (coloca los autómatas en su estado inicial).

El alfabeto entre niveles de supervisión es completamente disjunto, por lo que la ejecución de una transición *vocal* en un nivel superior de supervisión implica que por el canal de comunicación se "envíe" una secuencia de transiciones controlables a habilitar en el supervisor inmediato inferior. Si se trata de una estructura SOCC esto garantiza que al enviar dicha secuencia de transiciones controlables, el *camino* habilitado en el nivel inferior llevará únicamente al estado vocal deseado (sin pasar por ningún otro estado vocal). Dicho con otras palabras, existe una garantía de que lo estipulado en el supervisor superior podrá realizarse en el nivel inferior, sin que pase un evento indeseado antes. Estas cadenas de transiciones controlables deben de ser obtenidas en el proceso de diseño, antes de su implementación, con ayuda de los algoritmos de TCT.

Todas las transiciones controlables necesitan ser previamente habilitadas antes de poder ser ejecutadas en un estado. Solamente en el nivel de mayor jerarquía, que corresponde al *procedimiento de receta*, las transiciones controlables implican ejecuciones porque en este nivel es donde se lleva a cabo la decisión y el control de las tareas.

Las reglas de traducción dependen del nivel de supervisión en que se encuentre, para facilitar la explicación supondremos dos niveles de supervisión. Si se tuvieran más niveles de supervisión, en los niveles intermedios (es decir que son abstracciones de un nivel más bajo pero a la vez se forma una abstracción de ellos y se convierten en el nivel bajo para ésta) basta con aplicar primero las reglas de traducción para los niveles altos y luego al resultado aplicar las reglas para niveles bajos.

3.3.2. Representación en Diagrama de Escalera (LD)

En este caso se presenta el esquema de traducción a lenguaje de escalera, sin embargo la lógica empleada puede ser perfectamente representada en texto estructurado, utilizando los mismos elementos de ST mostrados en la sección 3.2. Los principios de la traducción son muy similares a los expuestos en la sección 3.1.

Se considera la estructura formal de una supervisor vocalizado representado con LD:

$$\bar{L} := (C, \Pi, \gamma, c_0, 2^S, 2^R)$$

Donde:

- C y Π son conjuntos de contactos normalmente abiertos con variables asociadas de tipo booleano, para modelar los estados (contacto de estado) y a las transiciones (contacto de transición) respectivamente. Existen los mapeos $\Psi : Y \rightarrow C$ y $\Phi : \Sigma \rightarrow \Pi$ que relacionan a éstos conjuntos de contactos normalmente abiertos con el conjunto de estados y transiciones. Los mapeos inversos $\Psi^{-1} : Y \rightarrow Q$ y $\Phi^{-1} : \Pi \rightarrow \Sigma$ también están definidos.

  $\gamma : C \times \Pi \rightarrow 2^S \times 2^R$ es la función parcial de transición que refleja la función parcial δ_S.

- y_0 es el estado inicial.

- 2^S y 2^R son conjuntos de conjuntos de bobinas de enganche y bobinas de liberación respectivamente, empleadas para cerrar o abrir los contactos normalmente abiertos que modelan los estados y las transiciones (modificando el estado de las variables booleanas asociadas a ellos).

El morfismo de δ_S a γ se construye como sigue: dado un supervisor con $y \in Y$ y $\sigma \in \Sigma$, si $\delta_S(y, \sigma)$ está definida, entonces $\gamma(\Psi(y), \Phi(\sigma))$ debe también estar definida. La implementación de la función γ dependerá del nivel de supervisión y de la naturaleza de las transiciones.

En el nivel inferior, cada **transición controlable** se desdobla en:

1. Una variable booleana asociada a un contacto normalmente abierto (el supervisor del nivel superior será el encargado de cerrar este contacto al enviar las cadenas de habilitación).

2. Una variable booleana que hace referencia a la transición controlable y que se asocia a una bobina de enganche.

Implementación en LD cuando $\sigma \in \Sigma_c$ en el nivel inferior

La función $\gamma(\Psi(y), \Phi(\sigma))$ está definida de la siguiente manera: $\gamma(c, \pi) = (\{s, s', s''\}, \{r, R'\})$ con $\tilde{c} = \Psi(\delta_S(y, \sigma))$. Donde s representa la bobina de enganche asociada al contacto de estado $\tilde{c} \in C$ (i.e el estado siguiente en el supervisor). s' está relacionada con la ejecución de la transición controlable y la bobina de enganche s'' es empleada para poner en verdadero la variable booleana relacionada con la salida vocal del estado. r representa la bobina de liberación asociada al contacto de estado $y \in Y$ (i.e estado presente en el supervisor) y R' es un conjunto de bobinas de liberación, que se emplea en estados vocalizados (símbolo de salida no silencioso) y que es utilizado para resetear todas las variables booleanas asociadas al conjunto de transiciones controlables de habilitación que fueron enviadas previamente por el supervisor de nivel superior para llegar al estado vocal presente. Si se trata de un evento vocal incontrolable no se mandaron cadenas para llegar a él y por lo tanto, el conjunto R' es vacío. Este conjunto también será vacío para el caso de estados cuyo símbolo de salida es silencioso.

La Figura 3.6 muestra un ejemplo de traducción para un estado no vocal del cual salen transiciones controlables.

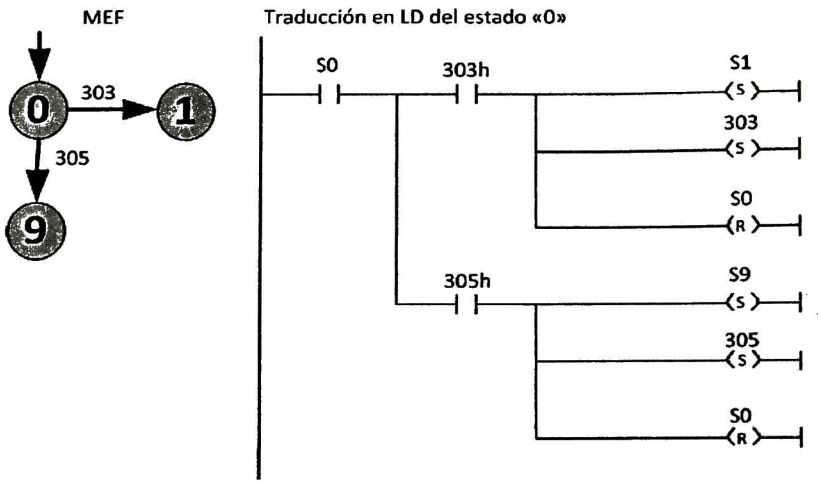


Figura 3.6: Ejemplo de traducción de transiciones controlables

Si un estado es *vocal* se debe *generar* el evento vocal en el estado mediante una bobina de enganche. Además en los eventos vocales se debe resetear (mediante bobinas de liberación) todas las variables asociadas a las cadenas de transiciones controlables de habilitación que fueron enviadas previamente por el supervisor de nivel superior para llegar al estado vocal presente. La Figura 3.7 muestra un ejemplo de traducción de un *estado vocal*. En este caso, la secuencia de transiciones controlables de habilitación enviada por el supervisor del nivel superior para llegar a este estado fue "317-311-315" es por eso que se resetean dichas transiciones al llegar al estado vocal 8. También se observa que se genera el evento vocal "341". Si en el nivel superior se trata de un evento vocal incontrolable entonces no se mandaron cadenas para llegar a él.

Implementación en LD cuando $\sigma \in \Sigma_u$ en el nivel inferior

En este caso la función $\gamma(\Psi(y), \Phi(\sigma))$ está definida de la siguiente manera: $\gamma(c, \pi) = (\{s, s', \{r, r', R''\}$ con $\tilde{c} = \Psi(\delta_S(y, \sigma))$. Donde s representa la bobina de enganche asociada al contacto de estado $\tilde{c} \in C$ (i.e el estado siguiente en el supervisor). s' está relacionada con la variable booleana asociada a la salida vocal del estado, definida por la función ω del generador de Moore. r representa la bobina de liberación asociada al contacto de estado $y \in Y$ (i.e estado presente en el supervisor), r' es la bobina de liberación que resetea la variable booleana asociada con $\sigma \in \Sigma_u$ y R' es un conjunto de bobinas de liberación empleado en estados vocalizados de igual manera que en el caso cuando $\sigma \in \Sigma_c$.

La Figura 3.8 muestra un ejemplo de traducción de un estado del cual salen transiciones incontrolables.

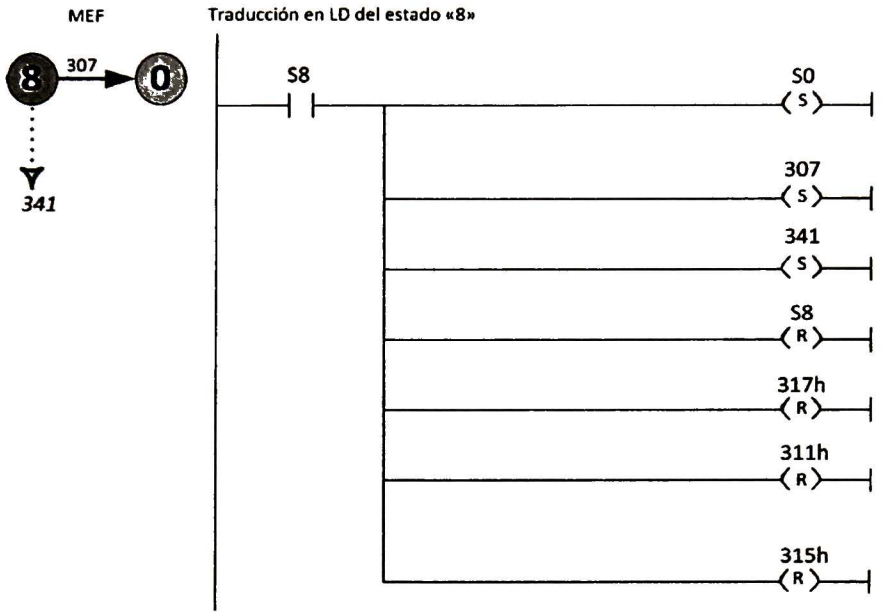


Figura 3.7: Ejemplo de traducción de un estado vocal

La Figura 3.9 muestra el caso en que de un estado salgan transiciones controlables e incontrolables a la vez, verificando secuencialmente si se ha presentado cada una de las transiciones incontrolables, de ser así se pasa al estado correspondiente indicado por la función δ, y en caso contrario se ejecuta la controlable.

Implementación en LD en el nivel superior

En el nivel superior, en cada estado se coloca en alto (mediante bobinas de enganche) la secuencia de transiciones controlables para alcanzar el estado vocal deseado. Estas cadenas deben ser obtenidas en la etapa de diseño como se mencionó.

Además en cada estado se espera como entrada (mediante un contacto normalmente abierto) que se presente el evento vocal correspondiente (al que se llegó con la cadena que se envió en el estado anterior).

Es decir, la función $\gamma(\Psi(y), \Phi(\sigma))$ está definida de la siguiente manera: $\gamma(c, \pi) = (\{s, S'\}, \{r, r'\})$ con $\tilde{c} = \Psi(\delta_S(y, \sigma))$. Donde s representa la bobina de enganche asociada al contacto de estado $\tilde{c} \in C$ (i.e el estado siguiente en el supervisor). S' es el conjunto de bobinas de enganche relacionado con las cadenas de transiciones controlables de habilitación del canal de comunicación hacia el nivel bajo. r se utiliza para resetear el estado actual y r' coloca en bajo la variable asociada con el evento vocal recién ocurrido.

En el apéndice B se muestra un ejemplo de implementación en diagrama de escalera para un esquema de supervisión de dos niveles.

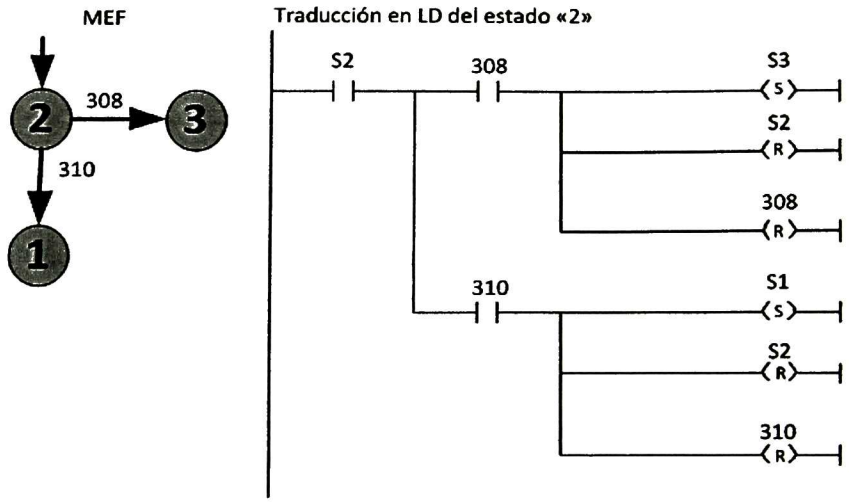


Figura 3.8: Ejemplo de traducción de transiciones incontrolables

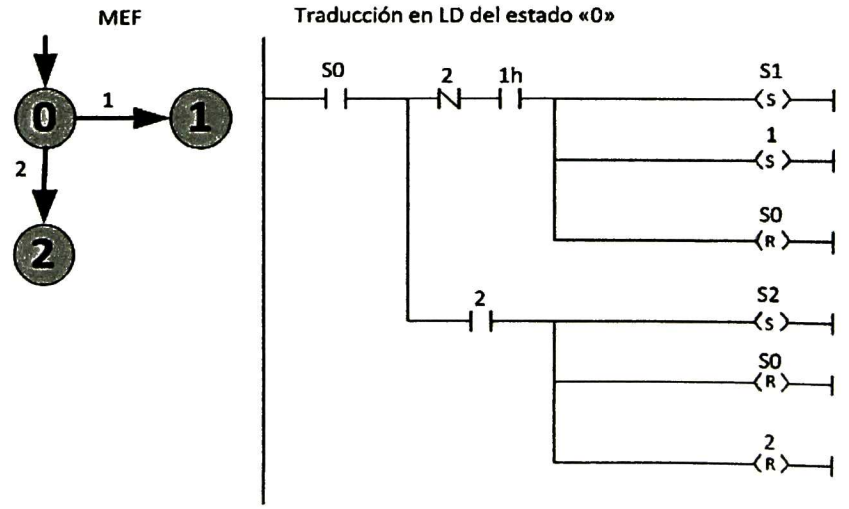


Figura 3.9: Ejemplo de traducción de transiciones controlables e incontrolables

Capítulo 4

Prototipo de Sistema Automatizado de Manufactura

En esta sección se describe el prototipo de SAM construido en Lego®, se detalla cada una de las secciones que lo conforma. Se explican los 18 componentes elementales con los que cuenta: la finalidad de cada uno, su modelo en MEF (se definen las transiciones relacionadas con cada elemento) y la implementación correspondiente en diagrama de escalera. Esta implementación corresponde a lo que se le denominó *capa de comunicación*, que es la capa I/0 entre el sistema a controlar y el dispositivo de control (PLC Siemens Simatic S7-300). También se presenta la interfaz electrónica destinada al acondicionamiento de señales, su configuración de pines y sus componentes. Finalmente se muestran los modelos de activos, de control de procedimientos e híbrido del SAM.

4.1. Descripción del prototipo

El prototipo de sistema automatizado de manufactura (SAM) construido en Lego®es un esquema típico de configuración en manufactura, constituido por dos despachadores de materia prima, dos estaciones de trabajo en las que se simula el procesamiento de la materia prima y la salida del producto terminado y una banda transportadora que une los despachadores con las estaciones de trabajo. En la Figura 4.1 se muestra su estructura.

El sistema de la banda transportadora consiste de cuatro bandas que forman un rectángulo, un par de motores impulsa el movimiento de estas bandas mediante un sistema de engranes. En este trabajo se considera a la banda en movimiento continuo sin embargo el prototipo y su interfaz electrónica están capacitados para realizar control sobre los motores de la banda. Este SAM fue construido en Lego®ya que es una herramienta práctica y económica para construir el prototipo.

4.1.1. Despachadores

Las unidades de materia prima están representadas por bloques individuales de Lego®y solamente hay dos tipos de materia prima (tipo A representada por bloques

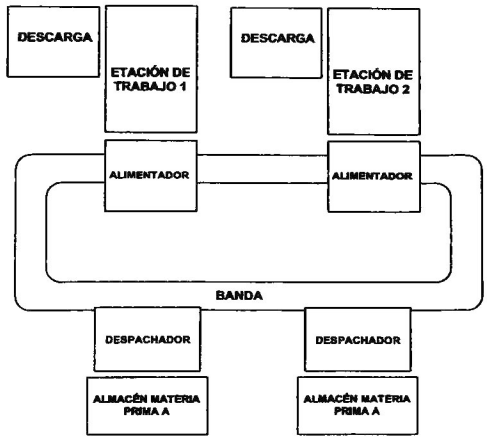


Figura 4.1: Diagrama a bloques del SAM

Lego®color café y tipo B simbolizada por bloques color blanco). La Figura 4.2 muestra los despachadores de materia prima, éstos están formados por una barra retráctil que es desplazada hacia adelante y hacia atrás mediante el giro de un engrane que es movido por un motor por medio de un sistema de banda y polea. Este movimiento permite que un bloque (materia prima) se deslice de la rampa al canal de la cremallera y sea depositada sobre la banda trasportadora.

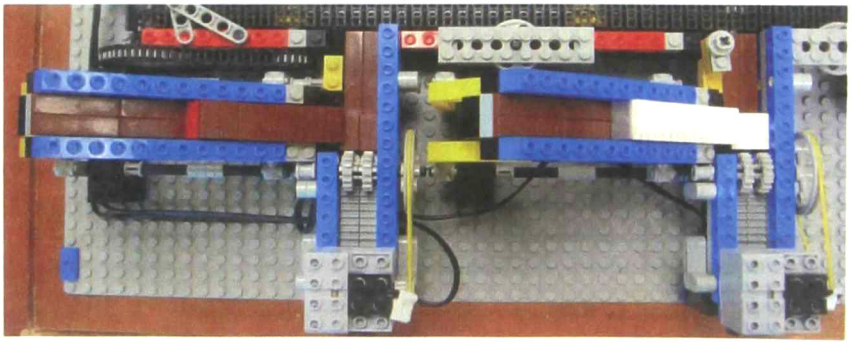


Figura 4.2: Despachadores de Materia Prima

4.1.2. Estaciones de Trabajo

Cada estación de trabajo está constituida por un **alimentador** de materia prima que introduce los bloques de Lego®procedentes de la banda hacia el área de trabajo de la estación, cuenta también con una sección de **maquinado y entrega** que simula el procesamiento de la materia prima que se introdujo en la estación (mediante el encendido

de un LED) y entrega el producto terminado mediante un sistema muy similar al que se encuentra en los despachadores (entrega los bloques introducidos previamente en la estación). La Figura 4.3 muestra una de las estaciones de trabajo.

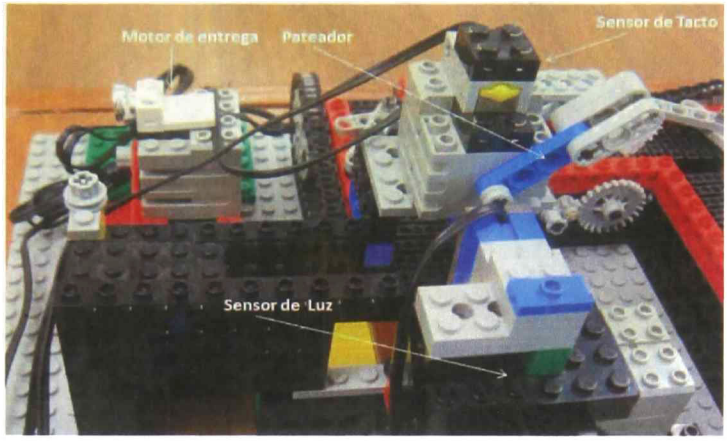


Figura 4.3: Estación de Trabajo

El alimentador está compuesto por un sensor de luz, un sensor de tacto y un motor; éste se encuentra colocado sobre el sistema de bandas y frente a la sección de maquinado y entrega de producto terminado. El sensor de luz detecta el color del bloque que pasa sobre la banda (color café o blanco únicamente aunque es posible configurarlo para reconocer más colores) y entonces la pieza se introduce al proceso por medio de un pateador que está unido al eje del motor. Al girar el brazo de éste se presiona el sensor de tacto que indica que el pateador ha dado una vuelta. El maquinado de la pieza únicamente es simulado. El sistema de salida de las piezas (producto final) consiste de un engrane (el cual es movido por un motor mediante un sistema de banda y polea) que mueve una cremallera que empuja las piezas hacia afuera de la máquina.

De esta manera cada estación de trabajo está capacitada para identificar el tipo de materia prima que se encuentra en la banda mediante el sensor de luz, introducir los bloques en el orden deseado mediante el pateador (informando mediante el sensor de tacto cada vez que se introduce una pieza) para después simular que se manufacturan y entregan productos distintos dependiendo de la materia prima involucrada.

En la Figura 4.4se muestra el SAM en su totalidad.

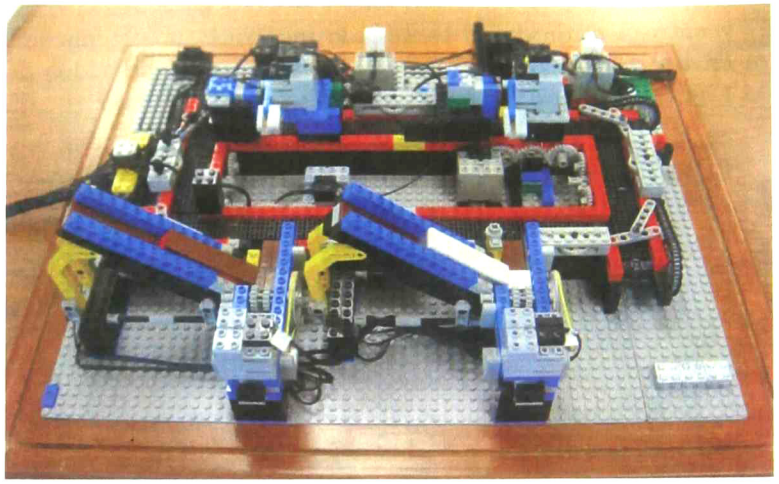


Figura 4.4: SAM construido en Lego®

4.2. Interfaz electrónica

4.2.1. Descripción

La Figura 4.5 muestra la interfaz electrónica destinada al acondicionamiento de señales y a la comunicación entre el SAM Lego®y el PLC Siemens Simatic S7-300. La tarjeta se desarrolló con el software Menthor Graphics. El esquemático, diagramas de colocación de componentes y ruteo se encuentran en el apéndice A. La tarjeta admite una alimentación de 5V a 18V DC sin embargo el funcionamiento adecuado se obtiene al alimentar con 10V (introducir un mayor voltaje implicaría una mayor velocidad en el movimiento de la banda, sin embargo también produce cambios en la lectura de los sensores de luz por lo que habría que recalibrar si se modifica el voltaje de alimentación).

La interfaz electrónica recibe como entrada las señales de control que provienen del PLC, las adecúa y las manda hacia los motores del SAM. También recibe como entrada las señales que provienen de los sensores de luz y de tacto del SAM, las acondiciona y las envía al PLC. Todas las señales que entran al conector DB25 etiquetado como "J8 " corresponden a las señales que son enviadas o recibidas del PLC (cable de la tarjeta al PLC). El conector etiquetado como "J9" en la tarjeta corresponde a todas las señales que se envían hacia el SAM Lego®(cable de la tarjeta al SAM Lego®).

4.2.2. Conectores y su Configuración de pines

En la Tabla 4.1 se presenta la configuración de pines de los conectores J8 y J9 de la tarjeta. Se muestra el pin del PLC al que está conectado, la señal Gxx que llega del PLC a la tarjeta y se convierte en la señal Jyy que va de la tarjeta hacia el SAM, donde

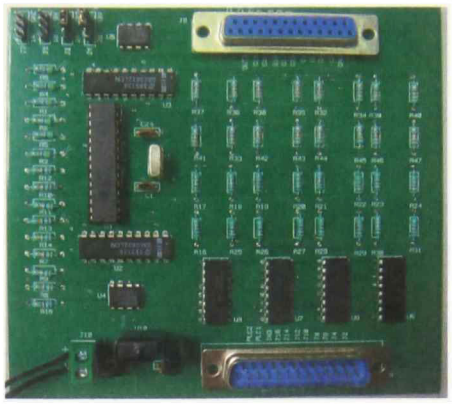


Figura 4.5: Interfaz Electrónica

yy y *xx* es el número del pin del conector correspondiente. Las entradas G13-G16 se encuentran desocupadas, estas entradas van hacia un manejador para motor de 4 canales (L293B) y las salidas de este componente van hacia los pines J1-J4. Es decir, estas entradas están destinadas al control de 2 motores "extras Pudiera ser que en algún momento se decida controlar el movimiento de la banda transportadora, entonces bastará conectar los motores correspondientes a estas entradas y colocar los jumpers J2, J3, J4 y J5 en su posición adecuada (ver esquemático de la tarjeta en el anexo A).

La posición estándar de los jumpers hace que la banda transportadora gire constantemente, nótese que se puede retirar el jumper J2 para hacer que la banda detenga su movimiento sin necesidad de quitar la alimentación.

4.2.3. Lista de Componentes

La lista de materiales empleados en la interfaz electrónica, así como su descripción, cantidad por tarjeta y costo en pesos al día 04/09/2010 se muestra en la Tabla 4.2.

Tabla 4.1: Configuración de pines de los conectores de la interfaz electrónica

Pin del PLC (Salidas)	Nombre de la señal (conector J8 "Al PLC")	Nombre de la señal (Conector J9 "Al Lego")	Descripción
A 125.7	G1	J15	Motor Despachador Materia Prima B "Mueve barra retráctil hacia atrás
A 125.6	G2	J16	Motor Despachador Materia Prima B "Mueve barra retráctil hacia adelante"
A 125.5	G3	J14	Motor Despachador Materia Prima A "Mueve barra retráctil hacia adelante"
A 125.4	G4	J13	Motor Despachador Materia Prima A "Mueve barra retráctil hacia atrás"
A 124.4	G5	J11	Motor para el maquinado en Estación 1 "Regresa barra retráctil"
A 124.5	G6	J12	Motor para el maquinado en Estación 1 "Empuja barra retráctil"
A 125.1	G7	J10	Motor para el maquinado en Estación 2 "Empuja barra retráctil"
A 125.0	G8	J9	Motor para el maquinado en Estación 2 "Regresa barra retráctil"
A 124.3	G9	J7	Motor Alimentador de Materia Prima en Estación 1 "Gira el pateador para introducir bloque"
A 124.2	G10	J8	Motor Alimentador de Materia Prima en Estación 1 "Gira el pateador en sentido inverso"
A 124.0	G11	J6	Motor Alimentador de Materia Prima en Estación 2 "Gira el pateador en sentido inverso"
A 124.1	G12	J5	Motor Alimentador de Materia Prima en Estación 2 "Gira el pateador para introducir bloque"
	G13	J1	
	G14	J2	Pines libres para manejar dos motores "extras"
	G15	J3	
	G16	J4	
Pin del PLC (Entradas)	**Nombre de la señal (conector J8 "Al PLC")**	**Nombre de la señal (Conector J9 "Al Lego")**	**Descripcion**
PEW 128	OUT1	IN1	Entrada Sensor Luz Estación 1
PEW 130	OUT3	IN3	Entrada Sensor Luz Estación 2
PEW 132	PLC1	PLC1	Entrada Sensor Tacto Estación 1
PEW 134	PLC2	PLC2	Entrada Sensor Tacto Estación 2

Tabla 4.2: Lista de Materiales de la Interfaz Electrónica

Componente	Etiqueta	Descripción	Cantidad	Precio
DAC0832LCN	U2, U3	DAC 8 bits	2	120.00 MXN
L293B	U6, U7, U8, U9	Driver push-pull de 4 canales	4	26.00 MXN
LM358N	U4, U5	Amplificador operacional lineal doble de bajo poder encapsulado N/626.	2	6.00 MXN
LM7805	U10	Regulador de voltaje a 5V	1	6.00 MXN
PIC16F873A	U1	Microcontrolador PIC, basado en tecnología CMOS de 8 bits.	1	120.00 MXN
Cristal 4Mhz	X1	Cristal de cuarzo con frecuencia de 4 MHz y carga capacitiva de 12 a 32 pF.	1	15.00 MXN
Cap. 22uF	C1,C2	Capacitor cerámico de disco de 22 uF (micro Faradios)	2	3.00 MXN
Conector	J10	Conector para la alimentación de 10V	1	10.00 MXN
R1.2K	R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28, R29, R30, R31	Resistencia de película de carbón de 1.2K Ohm a 1/4 de Watt	16	0.50 MXN
R1K	R15	Resistencia de película de carbón de 1K Ohm a 1/4 de Watt	1	0.50 MXN
R3.3K	R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42, R43, R44, R45, R46, R47	Resistencia de película de carbón de 3.3K Ohm a 1/4 de Watt	16	0.50 MXN
R4.7K	R1, R2, R3, R4, R5, R6, R12	Resistencia de película de carbón de 4.7K Ohm a 1/4 de Watt	6	0.50 MXN
R10K	R7, R8, R9, R10, R11, R13, R14	Resistencia de película de carbón de 10K Ohm a 1/4 de Watt	8	0.50 MXN
Jumpers	J2, J3, J4, J5	Puente eléctrico para tira de pines sencilla	4	2.00 MXN
Base para CI	N/A	Base para circuito integrado de 28 pines o agujas, para montar o desmontar un circuito integrado sin tener la necesidad de soldarlo.	1	5.00 MXN
Jack DB25	J8	Conector hembra (jack) DB25 para computadora. Con agujas para soldar, sin cubierta.	1	9.00 MXN
Plug DB25	J9	Conector macho (plug) DB25 para computadora. Con agujas para soldar, sin cubierta.	1	8.00 MXN
		Costo Aproximado de Componentes por Tarjeta		**566.50 MXN**

4.3. Ensamble final

En la Figura 4.6 se muestra el esquema de comunicación entre dispositivos. En el software Step7 se programan los controladores y por medio del puerto serie se descargan al PLC Siemens Simatic S7-300, las señales de control provenientes del PLC llegan a la interfaz electrónica por un cable con conector DB25 macho, se adecúan las señales y se envían al SAM por otro cable con conector DB25 hembra. La información de los sensores del SAM viaja por el mismo cable de conector DB25 hembra, pasa por la interfaz y de ahí al PLC para que éste determine la acción de control.

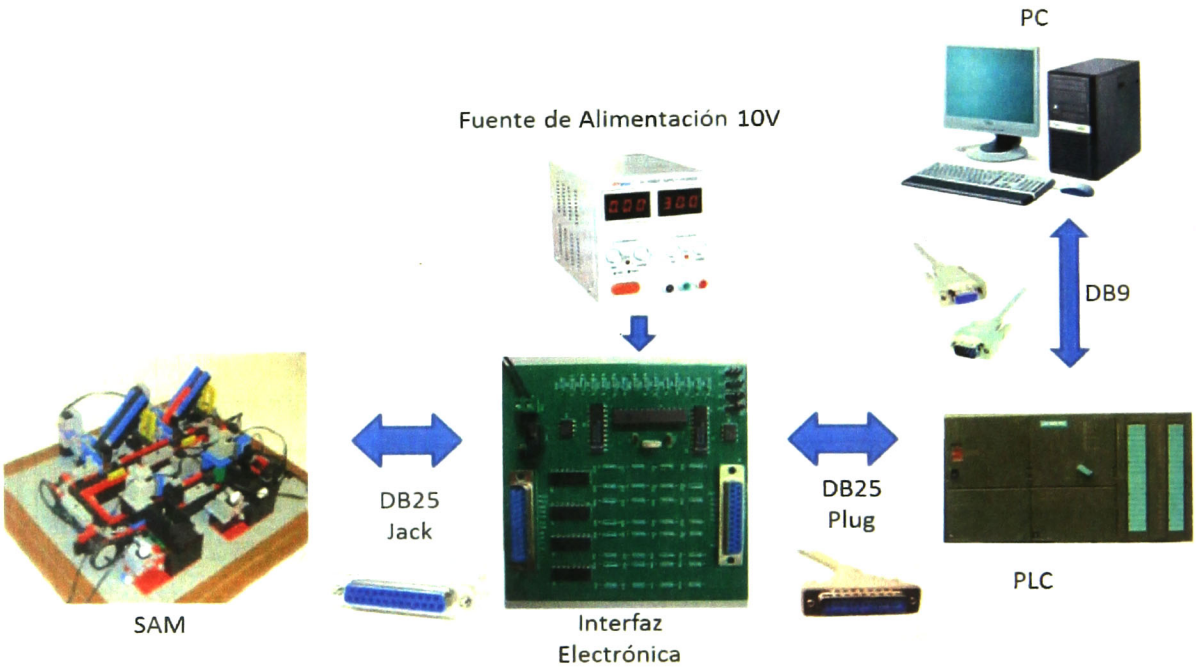


Figura 4.6: Comunicación entre dispositivos

El prototipo resultante es un sistema con complejidad suficiente para aplicar estrategias de control de interés. Así mismo, las ventajas de tener el prototipo en Lego®es que se pueden construir configuraciones más complejas simplemente reproduciendo más elementos como los descritos anteriormente o introduciendo nuevos equipos al SAM.

4.4. Componentes elementales

El SAM Lego®está formado por 18 componentes elementales, los cuales se muestran en la Tabla 4.3.

Tabla 4.3: Componentes Elementales del SAM

Componentes Elementales			
Sección	**Componente**	**Tipo**	**Descripción**
Despachadores	M1	Motor	Motor Despachador de Materia Prima A
	M2	Motor	Motor Despachador de Materia Prima B
	TA	Temporizador	Temporizador para el Despachador de Materia Prima A
	TB	Temporizador	Temporizador para el despachador de Materia Prima B
Alimentadores	M3	Motor	Motor Alimentador de Materia Prima en Estación 1
	M4	Motor	Motor Alimentador de Materia Prima en Estación 2
	RE1	Retardo	Retardo para el Alimentador en Estación 1
	RE2	Retardo	Retardo para el Alimentador en Estación 2
	SL1	Sensor	Sensor de Luz en Estación 1
	SL2	Sensor	Sensor de Luz en Estación 2
	ST1	Sensor	Sensor de Tacto en Estación 1
	ST2	Sensor	Sensor de Tacto en Estación 2
Maquinado y entrega	M5	Motor	Motor para el Maquinado en Estación 1
	M6	Motor	Motor para el Maquinado en Estación 2
	TE1	Temporizador	Temporizador para el Maquinado en la Estación 1
	TE2	Temporizador	Temporizador para el Maquinado en la Estación 2
	ME1	Simulación	Máquina de la Estación 1
	ME2	Simulación	Máquina de la Estación 2

Cada **despachador** involucra dos componentes elementales: un motor y un temporizador. Para que los despachadores funcionen debidamente se debe seguir el siguiente procedimiento, que puede ser modelado por una o varias especificaciones, se debe encender el motor de manera que éste desplace la barra hacia adelante para introducir un bloque de materia prima, enseguida se inicia el temporizador y cuando éste cuenta 1 segundo se debe detener el motor y accionarlo hacia la dirección opuesta e iniciar el temporizador nuevamente, cuando éste cuenta 1 segundo se debe detener el motor puesto que la barra ya ha regresado a su posición inicial. Es necesario utilizar el temporizador puesto que no se cuenta con sensores en los extremos de la barra desplazadora que nos indique cuando ésta ha llegado a la posición adecuada.

La sección de **alimentación** de materia prima de cada estación de trabajo involucra 4 componentes elementales: un sensor de tacto, un sensor de luz, un motor y un temporizador. El sensor de luz detecta el color del bloque (tipo de materia prima) y si es deseado se acciona el motor para hacer que gire el pateador e introducir el bloque a la estación de trabajo. Cuando el pateador realiza una vuelta completa se presiona el sensor de tacto para indicar que ya se ha terminado de introducir una pieza de materia prima. El temporizador es utilizado para considerar el tiempo que tarda en viajar la materia prima desde que fue detectada por el sensor de luz hasta que llega a la posición del pateador, este retardo depende de la velocidad de la banda y se encuentra calibrado para cuando se alimentan con 10V el sistema de bandas, sin embargo puede re-calibrarse fácilmente.

El área de **maquinado y entrega** de cada estación de trabajo involucra 3 componentes elementales, un componente virtual para simular el procesamiento de la materia prima, un temporizador y un motor. El sistema de entrega de producto terminado es semejante

al que se encuentra en los despachadores.

4.4.1. Lista de Transiciones relacionadas con los componentes elementales

En la Tabla 4.4 se muestra la lista de las transiciones relacionadas con los componentes elementales; se describe su significado, si se trata de una transición controlable o incontrolable, el tipo de variable que es en el PLC y la dirección de memoria en que se encuentra o el pin de salida al que está mapeada.

Tabla 4.4: Lista de transiciones de los Componentes Elementales

Equipo/Proceso	Símbolo	C/I	Dirección	Tipo	Comentario
Inicialización General	1000	C	M 10.0	BOOL	Inicialización de las máquinas (regresarlas a su estado inicial)
Despachar Materia Prima Tipo A	111	C	M 1.4	BOOL	Encender el motor hacia la derecha - (Motor Despachador Materia Prima A)
	113	C	M 1.5	BOOL	Encender el motor hacia la izquierda - (Motor Despachador Materia Prima A)
	115	C	M 1.6	BOOL	Detener el motor - (Motor Despachador Materia Prima A)
	116	I	M 2.0	BOOL	Terminar Temporizador A - (Despachador Materia Prima A)
	117	C	M 1.7	BOOL	Iniciar Temporizador A - (Despachador Materia Prima A)
Despachar Materia Prima Tipo B	211	C	M 2.1	BOOL	Encender el motor hacia la derecha - (Motor despachador Materia Prima B)
	213	C	M 2.2	BOOL	Encender el motor hacia la izquierda - (Motor despachador Materia Prima B)
	215	C	M 2.3	BOOL	Detener el motor - (Motor Despachador Materia Prima B)
	216	I	M 2.5	BOOL	Terminar Temporizador B - (Despachador Materia Prima B)
	217	C	M 2.4	BOOL	Iniciar Temporizador B - (Despachador Materia Prima B)
Alimentación de materia prima en Estación 1	307	C	M 3.3	BOOL	Se habilita el funcionamiento del Sensor de Luz - Estación 1
	308	I	M 3.4	BOOL	El sensor detecta una pieza café (tipo A) - Estación 1
	310	I	M 3.5	BOOL	El sensor detecta una pieza blanca(tipo B) - Estación 1
	311	C	M 3.6	BOOL	Encender motor hacia la derecha - Motor Alimentador Estación 1
	312	I	M 4.0	BOOL	Sensor de tacto presionado - Estación 1
	313	C	M 3.7	BOOL	Encender motor hacia la izquierda - Motor Alimentador Estación 1
	315	C	M 4.1	BOOL	Detener motor - Motor Alimentador Estación 1
	317	C	M 8.4	BOOL	Iniciar retardo del Pateador en Estación 1
	318	I	M 8.5	BOOL	Terminar retardo del Pateador en Estación 1
Alimentación de materia prima en Estación 2	407	C	M 4.6	BOOL	Se habilita el funcionamiento del Sensor de Luz - Estación 2
	408	I	M 4.7	BOOL	El sensor detecta una pieza café (tipo A) - Estación 2
	410	I	M 5.0	BOOL	El sensor detecta una pieza blanca (tipo B) - Estación 2
	411	C	M 5.1	BOOL	Encender motor hacia la derecha - Motor Alimentador Estación 2
	412	I	M 5.3	BOOL	Sensor de Tacto presionado - Estación 2
	413	C	M 5.2	BOOL	Encender motor hacia la izquierda - Motor Alimentador Estación 2
	415	C	M 5.4	BOOL	Detener motor - Motor Alimentador Estación 2
	417	C	M 8.6	BOOL	Iniciar retardo del Pateador en estación 2
	418	I	M 8.7	BOOL	Terminar retardo del Pateador en estación 2
Maquinado y entrega de producto en estación 1	506	I	M 6.5	BOOL	Termina el temporizador - Maquinado en Estación 1
	507	C	M 6.4	BOOL	Iniciar el temporizador - Maquinado en Estación 1
	508	I	M 6.0	BOOL	Termina el Maquinado - Estación 1
	509	C	M 5.7	BOOL	Iniciar Maquinado - Estación 1
	511	C	M 6.1	BOOL	Encender motor hacia la derecha - Motor Maquinado en Estación 1
	513	C	M 6.2	BOOL	Encender motor hacia la izquierda - Motor Maquinado en Estación 1
	515	C	M 6.3	BOOL	Detener motor - Motor Maquinado en Estación 1
Maquinado y entrega de producto en estación 1	606	I	M 7.6	BOOL	Termina el temporizador - Maquinado en Estación 2
	607	C	M 7.5	BOOL	Iniciar el temporizador - Maquinado en Estación 2
	608	I	M 7.1	BOOL	Termina el Maquinado - Estación 2
	609	C	M 7.0	BOOL	Iniciar Maquinado - Estación 2
	611	C	M 7.2	BOOL	Encender motor hacia la derecha - Motor Maquinado en Estación 2
	613	C	M 7.3	BOOL	Encender motor hacia la izquierda - Motor Maquinado en Estación 2
	615	C	M 7.4	BOOL	Detener motor - Motor Maquinado en Estación 2

Como se expondrá en la sección 4.4.2, cada componente elemental se implementó como una función en diagrama de escalera para así conformar la capa de comunicación entre el PLC y el SAM. La Tabla 4.5 muestra el listado de componentes elementales y el número de la función que le corresponde en la implementación. La Tabla 4.6 muestra las entradas y salidas del PLC y la Tabla 4.7 muestra el listado de variables que representa el estado

correspondiente de las MEF que modelan a los componentes elementales, éstos modelos se muestran en la siguiete sección 4.4.2. Toda la información de estas tablas se concentra en la "Tabla de Símbolos" de Step7.

Tabla 4.5: Funciones correspondientes a los Componentes Elementales

Motor Despachador A	FC 1	M1 Motor Despachador de Materia Prima A
Motor Despachador B	FC 2	M2 Motor Despachador de Materia Prima B
Motor Alimentador E1	FC 3	M3 Motor Alimentador de Materia Prima en Estación 1
Motor Alimentador E2	FC 4	M4 Motor Alimentador de Materia Prima en Estación 2
Motor Maquinado E1	FC 5	M5 Motor para el Maquinado en Estación 1
Motor Maquinado E2	FC 6	M6 Motor para el Maquinado en Estación 2
Temporizador A	FC 7	TA Temporizador para el Despachador de Materia Prima A
Temporizador B	FC 8	TB Temporizador para el Despachador de Materia Prima B
TemporizadorMaquinado E1	FC 9	TE1 Temporizador para el Maquinado en la Estación 1
TemporizadorMaquinado E2	FC 10	TE2 Temporizador para el Maquinado en la Estación 2
Retardo Alimentador E1	FC 11	RE1 Retardo para el Alimentador en Estación 1
Retardo Alimentador E2	FC 12	RE2 Retardo para el Alimentador en Estación 2
Sensor Luz E1	FC 13	SL1 Sensor de Luz en Estación 1
Sensor Luz E2	FC 14	SL2 Sensor de Luz en Estación 2
Sensor Tacto E1	FC 15	ST1 Sensor de Tacto en Estación 1
Sensor Tacto E2	FC 16	ST2 Sensor de Tacto en Estación 2
Maquina E1	FC 17	ME1 Máquina de la Estación 1
Maquina E2	FC 18	ME2 Máquina de la Estación 2

Tabla 4.6: Entradas y Salidas del PLC

Entrada / Salida	Nombre	Dirección	Tipo	Comentario
Entradas	Entrada Sensor Luz E1	PEW 128	INT	Entrada Sensor Luz E1
	Entrada Sensor Luz E2	PEW 130	INT	Entrada Sensor Luz E2
	Entrada Sensor Tacto E1	PEW 132	INT	Entrada Sensor Tacto E1
	Entrada Sensor Tacto E2	PEW 134	INT	Entrada Sensor Tacto E2
Salidas	M1 Adelante	A 125.5	BOOL	Salida - Motor Despachador Materia Prima A "Hacia Adelante"
	M1 Reversa	A 125.4	BOOL	Salida - Motor Despachador Materia Prima A "Hacia Atrás"
	M2 Adelante	A 125.6	BOOL	Salida - Motor Despachador Materia Prima B "Hacia Adelante"
	M2 Reversa	A 125.7	BOOL	Salida - Motor Despachador Materia Prima B "Hacia Atrás"
	M3 Adelante	A 124.3	BOOL	Salida - Motor Alimentador de Materia Prima en Estación 1 "Hacia adelante"
	M3 Reversa	A 124.2	BOOL	Salida - Motor Alimentador de Materia Prima en Estación 1 "Hacia atrás"
	M4 Adelante	A 124.1	BOOL	Salida - Motor Alimentador de Materia Prima en Estación 2 "Hacia adelante"
	M4 Reversa	A 124.0	BOOL	Salida - Motor Alimentador de Materia Prima en Estación 2 "Hacia atrás"
	M5 Adelante	A 124.5	BOOL	Salida - Motor para el Maquinado en Estación 1 "Hacia Adelante"
	M5 Reversa	A 124.4	BOOL	Salida - Motor para el Maquinado en Estación 1 "Hacia Atrás"
	M6 Adelante	A 125.1	BOOL	Salida - Motor para el Maquinado en Estación 2 "Hacia Adelante"
	M6 Reversa	A 125.0	BOOL	Salida - Motor para el Maquinado en Estación 2 "Hacia Atrás"

4.4.2. Modelo e Implementación de los Componentes Elementales

En esta sección se muestra el modelo matemático en MEF que corresponde a cada componente elemental así como su implementación en diagrama de escalera (LD)

Tabla 4.7: Variables para denotar los estados de las MEFs que modelan a los componentes elementales

Nombre	Dirección	Tipo	Comentario
M1_S0	M 37.3	BOOL	Estado 0 Motor Despachador Materia Prima A
M1_S1	M 37.4	BOOL	Estado 1 Motor Despachador Materia Prima A
M2_S0	M 37.5	BOOL	Estado 0 Motor Despachador Materia Prima B
M2_S1	M 37.6	BOOL	Estado 1 Motor Despachador Materia Prima B
M3_S0	M 43.1	BOOL	Estado 0 Motor Alimentador de Materia Prima en Estación 1
M3_S1	M 43.2	BOOL	Estado 1 Motor Alimentador de Materia Prima en Estación 1
M4_S0	M 43.3	BOOL	Estado 0 Motor Alimentador de Materia Prima en Estación 2
M4_S1	M 43.4	BOOL	Estado 1 Motor Alimentador de Materia Prima en Estación 2
M5_S0	M 37.7	BOOL	Estado 0 Motor Maquinado E1
M5_S1	M 38.0	BOOL	Estado 1 Motor Maquinado E1
M6_S0	M 38.1	BOOL	Estado 0 Motor Maquinado E2
M6_S1	M 38.2	BOOL	Estado 1 Motor Maquinado E2
TA_S0	M 43.5	BOOL	Estado 0 Temporizador para el Despachador de Materia Pprima A
TA_S1	M 43.6	BOOL	Estado 1 Temporizador para el Despachador de Materia Prima A
TB_S0	M 43.7	BOOL	Estado 0 Temporizador para el Despachador de Materia Prima B
TB_S1	M 44.0	BOOL	Estado 1 Temporizador para el Despachador de Materia Prima B
TE1_S0	M 44.1	BOOL	Estado 0 Temporizador para el Maquinado en la Estación 1
TE1_S1	M 44.2	BOOL	Estado 1 Temporizador para el Maquinado en la Estación 1
TE2_S0	M 44.3	BOOL	Estado 0 Temporizador para el Maquinado en la Estación 2
TE2_S1	M 44.4	BOOL	Estado 1 Temporizador para el Maquinado en la Estación 2
RE1_S0	M 44.5	BOOL	Estado 0 Retardo para el Alimentador en Estación 1
RE1_S1	M 44.6	BOOL	Estado 1 Retardo para el Alimentador en Estación 1
RE2_S0	M 44.7	BOOL	Estado 0 Retardo para el Alimentador en Estación 2
RE2_S1	M 45.0	BOOL	Estado 1 Retardo para el Alimentador en Estación 2
SL1_0	M 45.1	BOOL	Estado 0 Sensor de Luz en Estación 1
SL1_1	M 45.2	BOOL	Estado 1 Sensor de Luz en Estación 1
SL2_0	M 45.3	BOOL	Estado 0 Sensor de Luz en Estación 2
SL2_1	M 45.4	BOOL	Estado 1 Sensor de Luz en Estación 2
ST1_S0	M 45.5	BOOL	Estado 0 Sensor de Tacto en Estación 1
ST2_S0	M 45.6	BOOL	Estado 0 Sensor de Tacto en Estación 2
ME1_S0	M 45.7	BOOL	Estado 0 Máquina de la Estación 1
ME1_S1	M 46.0	BOOL	Estado 1 Máquina de la Estación 1
ME2_S0	M 46.1	BOOL	Estado 0 Máquina de la Estación 2
ME2_S1	M 46.2	BOOL	Estado 1 Máquina de la Estación 2

empleando el esquema de traducción mostrado en la sección 3.1. Cada componente elemental se implementó como una función en el Step7.

Motores

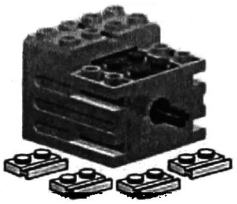


Figura 4.7: Motor MINDSTORMS®

En la Figura 4.7 se muestra un motor Lego®MINDSTORMS® . El SAM Lego®cuenta con 6 motores a controlar (M1-M6). La Figura 4.8 muestra el modelo en MEF y su correspondiente implementación en diagrama de escalera. Los motores M2-M6 fueron implementados con la misma lógica. M1 y M2 son los motores que se encargan de mover la cremallera para despachar la materia prima, M3 y M4 mueven los pateadores para alimentar materia prima a las estaciones de trabajo, M5 y M6 mueven la cremallera para sacar el producto terminado de las estaciones de trabajo.

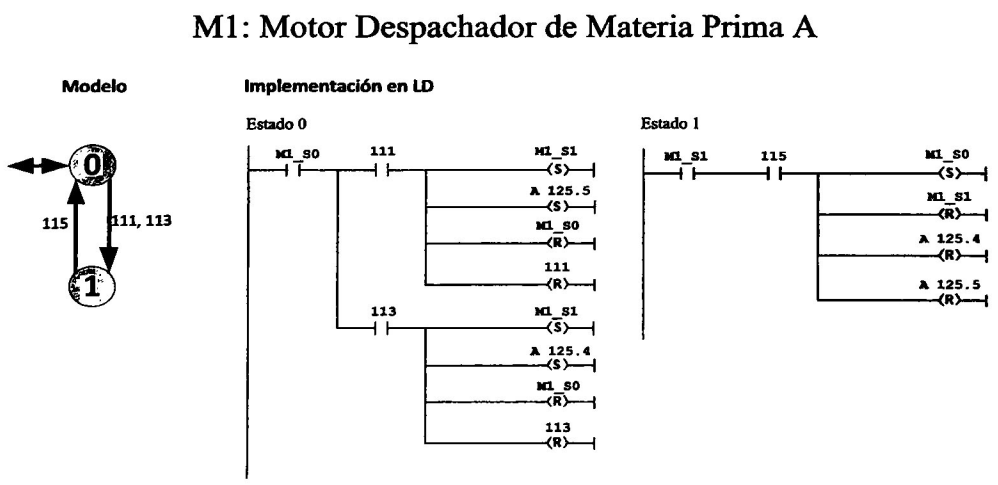


Figura 4.8: M1

Temporizadores

Se tienen 6 temporizadores: TA, TB, TE1, TE2, RE1 y RE2. Los dos primeros (TA y TB) son utilizados en los despachadores para indicar a los motores M1 y M2 respectivamente el tiempo que deben estar encendidos hacia una dirección dada. TE1 y TE2 son temporizadores empleados en las estaciones de trabajo para indicar a M5 y M6 (motores encargados de la entrega de producto terminado) el tiempo que deben estar encendidos hacia una dirección determinada (este tiempo se indica en la entrada "TW" de los temporizadores y está dada en milisegundos).Como ejemplo de implementación se muestra el temporizador TA en la Figura 4.9 ya que todos se implementaron empleando la misma lógica.

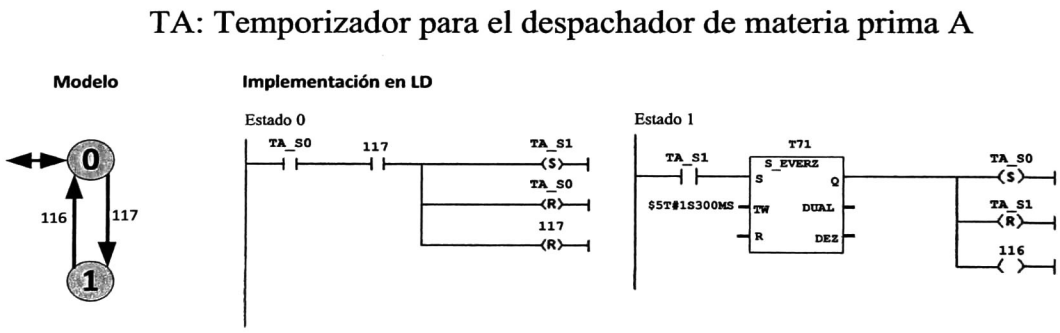


Figura 4.9: TA

Los retardos RE1 y RE2 son necesarios en las estaciones de trabajo, se requiere esperar cierto tiempo a partir de que el sensor de luz detectó una pieza requerida antes de que el pateador pueda introducirla a la estación. El tiempo que tarda en pasar la pieza de la ubicación del sensor de luz a donde se encuentra el pateador depende de la velocidad de la banda la cual está sujeta al voltaje de alimentación y otros factores (como el desgaste de los motores). El retardo actual se calculó tomando en cuenta una alimentación de 10 Volts y motores en condiciones óptimas, sin embargo es posible que este retardo requiera de calibración al momento de operar el SAM bajo otras condiciones.

Sensores de Luz y Tacto

El SAM cuenta con un par de sensores de luz: SL1 y SL2 (uno en cada estación de trabajo) para detectar el color del bloque de materia prima que pasará por el alimentador. La Figura 4.10 muestra un sensor de luz de Lego®MINDSTORMS® Como ejemplo, la Figura 4.11 muestra su modelos e implementación en diagrama de escalera, ambos sensores de luz fueron implementados de la misma manera.

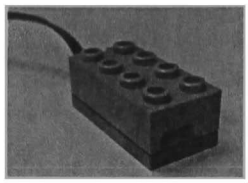


Figura 4.10: Sensor de Luz MINDSTORMS®

El sensor de luz emite un rayo de luz color rojo, el rayo incide sobre el bloque de materia prima y dependiendo del color absorbe parte de la radiación electromagnética y otra parte la refleja en el ángulo de incidencia, la luz reflejada es capturada por el sensor fotoeléctrico y produce cierto voltaje dependiendo de la intensidad de luz recibida, este voltaje pasa por la interfaz electrónica y lo envía al PLC por una de sus entradas en forma de un número entero.
Para el SL1, si el PLC recibe entre 13,000 y 13,800 unidades se trata de una pieza color café pero si detecta entre 14,200 y 15,000 será una pieza color blanco. Si el PLC recibe un número menor a 13,000 significa que no hay pieza alguna. Para el SL2, el rango que corresponde al color café es entre 11,300 y 12,200 unidades y para el color blanco entre 12,400 y 13,000 unidades. Si el PLC recibe un número menor a 11,300 significa que no hay pieza alguna. Es posible que estos datos requieran calibración dependiendo de la luminosidad de la habitación o si se cambia de color las piezas de materia prima.

Cada estación de trabajo cuenta con un sensor de tacto (ST1 y ST2) para indicar el momento en que se ha alimentado a la estación de trabajo. El pateador alimenta una pieza y al girar presiona el sensor de tacto. La Figura 4.12 muestra un sensor de tacto de Lego®MINDSTORMS® La Figura 4.13 muestra el modelo e implementación en diagrama de escalera del sensor ST1.

Maquinado virtual

El SAM cuenta con 2 señales virtuales o de simulación que aparentan el maquinado del producto. ME1 simula que se ha ensamblado un producto en la estación 1 y ME2 hace lo correspondiente en la estación 2. La Figura 4.14 muestra el modelo e implementación en diagrama de escalera de ME1.

4.4.3. Modelo de Equipo y de Procedimientos

Se tiene una arquitectura de coordinación modular de tres niveles, diseñada en trabajos anteriores [27], [28]. La capa más baja está dedicada al control del equipo, ejecutando *fases*, mientras que la capa superior recibe el *procedimiento de receta* (modelado como MEF) imponiendo los *procedimientos unitarios* y la secuencia de las *fases*.

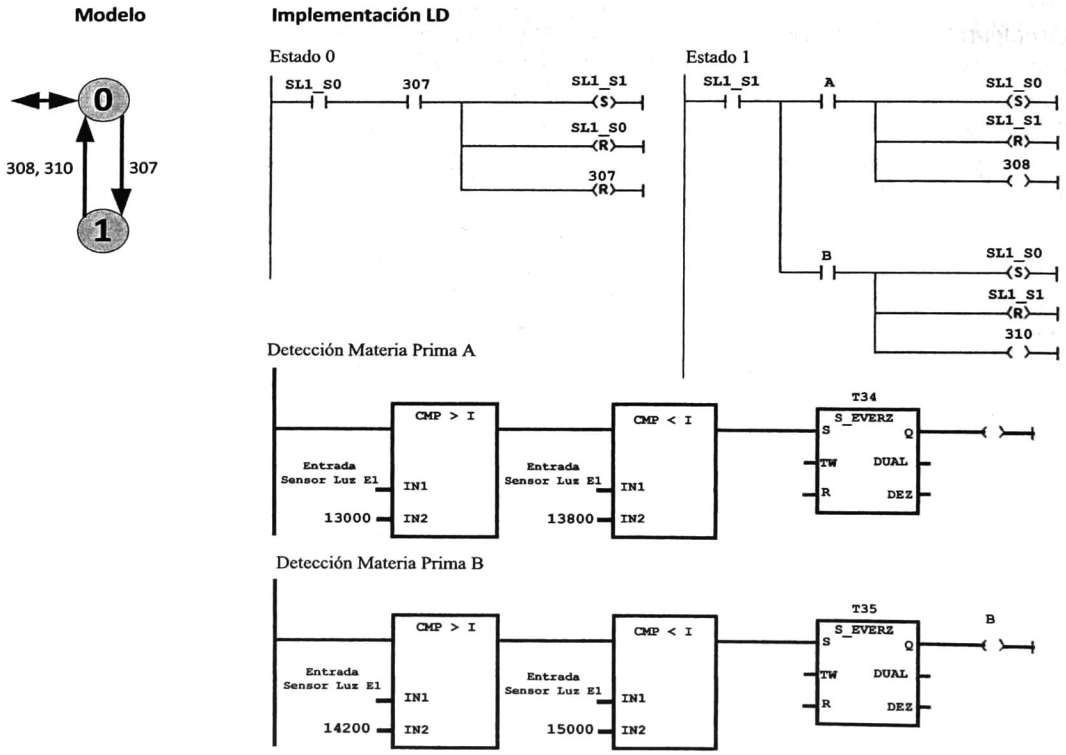


Figura 4.11: SL1

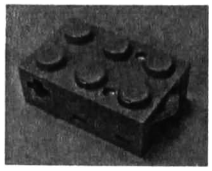


Figura 4.12: Sensor de Tacto MINDSTORMS®

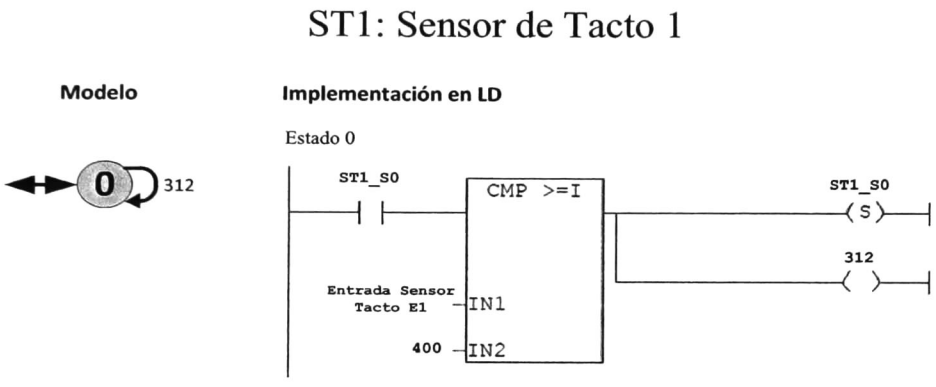


Figura 4.13: ST1

Es decir, a nivel *fase* y *procedimiento unitario* se sintetizan supervisores empleando el procedimiento descrito en [11]; se sincronizan los modelos de los componentes elementales y comportamientos causales, se modelan especificaciones que se sincronizan para formar una especificación monolítica y finalmente se obtiene el supremo sub-lenguaje controlable que cumple con todas las especificaciones planteadas para restringir el comportamiento de la planta, deshabilitando determinadas transiciones controlables. Se sintetiza un supervisor por cada módulo (i.e. *procedimientos unitarios* y *fases*).

En la capa de mayor jerarquía se encuentra la *receta* que consta de una secuencia de ejecuciones, en este nivel se establece el programa de producción y viene a ser una especie de controlador que comanda los niveles inferiores, habilitando o deshabilitando transiciones en los supervisores inferiores.

Se emplearon los estándares ISA88 [38] e ISA95 [39] para construir los modelos de equipo y de procedimiento.

El modelo de activos es un agrupamiento jerárquico de los activos físicos. El modelo de control de procedimientos establece las actividades que ejecutan los activos físicos para realizar tareas con un sentido de proceso. El resultado de combinar los equipos con los procedimientos es llamado modelo de proceso.

Aplicando los estándares mencionados, se obtienen los modelos de activos, de control de procedimientos e híbrido del SAM.

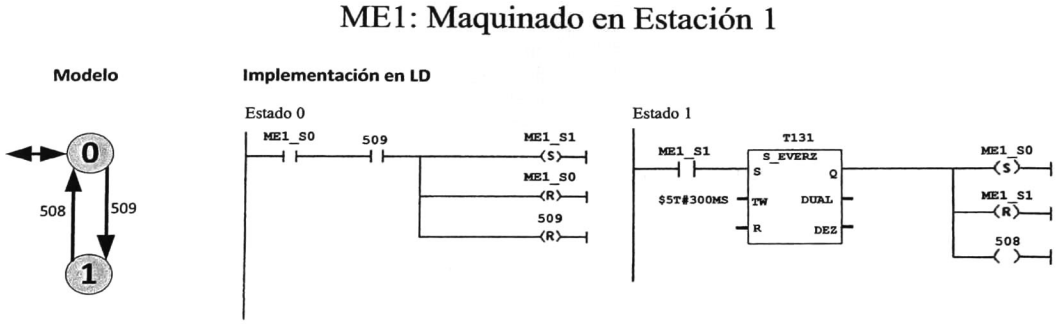


Figura 4.14: ME1

En la Tabla 4.8 se muestra el ordenamiento jerárquico-modular de los activos del SAM, es decir los componentes de cada módulo de equipo relacionados con el control del sistema. En la Tabla 4.9 se presenta el modelo de control de procedimientos. En la Figura 4.15, se muestra el modelo híbrido del SAM.

Tabla 4.8: Modelo de Activos

Modelo de Activos			
Celda de Proceso	**Unidad**	**Módulo de Equipo**	**Componentes Elementales**
Celda de Manufactura	Despachadores	Despachador de materia prima A	M1, TA
		Despachador de materia prima B	M2, TB
	Estación de Trabajo 1	Alimentador de materia prima	M3, RE1, SL1, ST1
		Maquinado y Entrega	M5, ME1, TE1
	Estación de Trabajo 2	Alimentador de materia prima	M4, RE2, SL2, ST2
		Maquinado y Entrega	M6, ME2, TE2

Tabla 4.9: Modelo de Control de Procedimientos

Modelo de Control de Procedimientos			
	Procedimiento Unitario	Fase de Equipo	Acción
Producir Producto	Despachar Materia Prima	Despachadores de materia prima A ó B	i. Encender M1/M2 hacia adelante el tiempo indicado por TA/TB y luego detener. ii. Encender M1/M2 hacia atrás el tiempo indicado por TA/TB y luego detener.
	Procesar Materia Prima en Estación de Trabajo 1	Alimentar Materia Prima A ó B	i. Encender SL1 y esperar que se detecte materia prima deseada. ii. Una vez detectada la materia prima deseada se deja pasar el tiempo indicado por RE1 para luego encender M3 hasta que se presione ST1.
		Maquinado y Entrega	i. Iniciar ME1. ii. Terminar ME1. iii. Encender M5 hacia adelante el tiempo indicado por TE1 y apagar. Iv. Encender M5 hacia atrás el tiempo indicado por TE1 y apagar.
	Procesar Materia Prima en Estación de Trabajo 2	Alimentar Materia Prima A ó B	i. Encender SL2 y esperar que se detecte materia prima deseada. ii. Una vez detectada la materia prima deseada se deja pasar el tiempo indicado por RE2 para luego encender M4 hasta que se presione ST2.
		Maquinado y Entrega	i. Iniciar ME2. ii. Terminar ME2. iii. Encender M6 hacia adelante el tiempo indicado por TE1 y apagar. Iv. Encender M6 hacia atrás el tiempo indicado por TE1 y apagar.

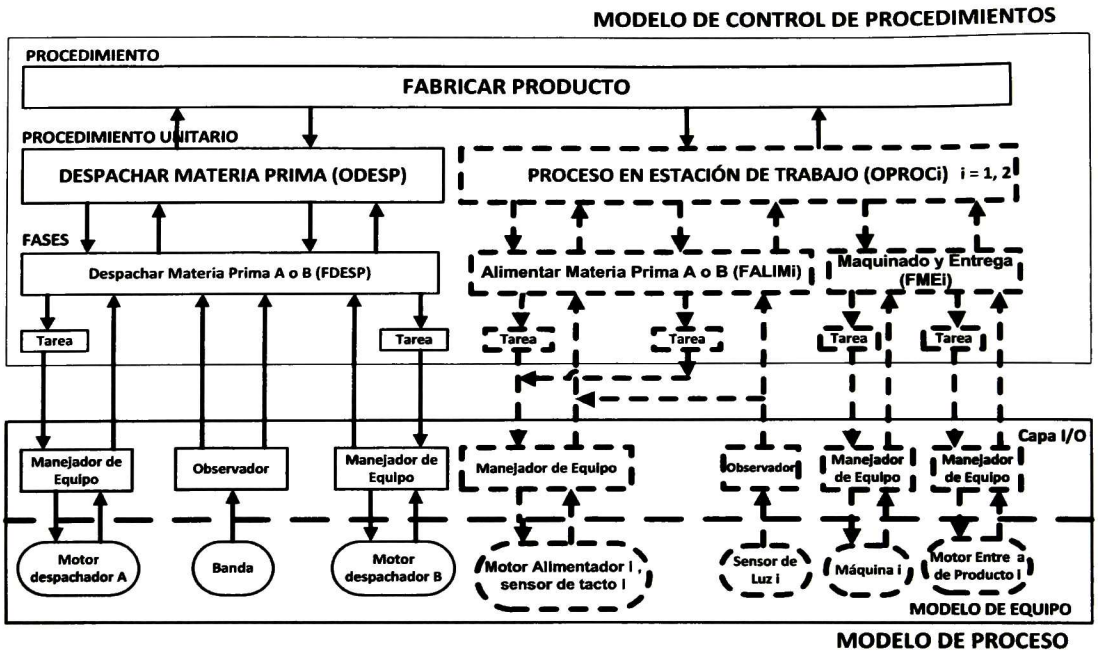


Figura 4.15: Modelo Híbrido del SAM

Capítulo 5

Aplicación de Control Jerárquico-Modular al caso de estudio

En esta sección se hará la síntesis de los supervisores de acuerdo con el modelo híbrido mostrado en la Figura 4.15, empleando la técnica de *vocalización* y la de *control por eventos imperativos*. Se implementaron estos supervisores siguiendo las reglas expuestas en la secciones 3.2 y 3.3.

A nivel *fase* se sintetizaron cinco supervisores dedicados a realizar tareas relacionadas con el equipo:

1. "Supervisor Despachador de Materia Prima" (FDESP),
2. "Supervisor Alimentar Materia Prima en Estación 1" (FALIM1),
3. "Supervisor Alimentar Materia Prima en Estación 2" (FALIM2),
4. "Supervisor Maquinado y Entrega de Producto en Estación 1" (FME1),
5. "Supervisor Maquinado y Entrega de Producto en Estación 2" (FME2).

Se sintetizaron tres supervisores de nivel *operación*, sin embargo su estructura es diferente dependiendo de la técnica empleada:

1. "Operación Despachar Materia Prima" (ODESP)
2. "Operación Proceso en Estación 1" (OPROC1)
3. "Operación Proceso en Estación 2" (OPROC2)

Se tienen tres bloques de *procedimiento de receta*:

1. "Receta Despachar Materia Prima" (RDESP)

2. "Receta Proceso en Estación 1" (RPROC1)

3. "Receta Proceso en Estación 2" (RPROC2)

A manera de ejemplo, las *recetas* diseñadas corresponden a la manufactura de cuatro productos: dos formados por materia prima tipo A + materia prima tipo B y el otro con los mismos materiales pero en orden invertido (Tipo B + Tipo A); es decir, se le está dando importancia al orden en que se alimenta la materia prima que conforma el producto.

De manera adicional a estos supervisores se implementó una función encargada de la inicialización de todos los supervisores (coloca los autómatas en su estado inicial) y otra que conjunta las tres recetas (esta máquina se encarga de iniciar todas las recetas para comenzar la producción).

La Tabla 4.4 muestra las transiciones relacionadas con los componentes elementales, sin embargo es necesario definir más transiciones para conseguir la estructura jerárquica, tanto para indicar el inicio o fin de tarea (Tabla 5.1) como los eventos vocales (Tabla 5.2).

Tabla 5.1: Transiciones empleadas para la inicialización y para indicar el inicio o fin de una tarea

Transición	Descripción
0	Inicialización de todas las máquinas en su estado inicial.
1	iniciar receta para Despachar Materia Prima.
2	Iniciar receta para Alimentar y Entregar Producto en la Estación 1.
3	Iniciar receta para Alimentar y Entregar Producto en la Estación 2.
4	Forzar inicio general (iniciar producción).
103	Iniciar fase de Despachar Materia Prima A
104	Finalizar la fase de Despachar Materia Prima A
203	Iniciar fase de Despachar Materia Prima B
204	Finalizar la fase de Despachar Materia Prima B
303	Iniciar fase de Alimentación de Materia Prima A Estación 1
304	Finalizar fase de Alimentación de Materia Prima A - Estación 1
305	Iniciar fase de Alimentación de Materia Prima B - Estación 1
306	Finalizar fase de Alimentación de Materia Prima B - Estación 1
403	Iniciar fase de Alimentación de Materia Prima A - Estación 2
404	Finalizar fase de Alimentación de Materia Prima A - Estación 2
405	Iniciar fase de Alimentación de Materia Prima B - Estación 2
406	Finalizar fase de Alimentación de Materia Prima B - Estación 2
503	Iniciar fase de Maquinado y Entrega de Producto Estación 2
504	Finalizar fase de Maquinado y Entrega de Producto Estación 1
603	Iniciar fase de Maquinado y Entrega de Producto Estación 2
604	Finalizar fase de Maquinado y Entrega de Producto Estación 2

5.1. Síntesis de Supervisores: Control por Eventos Imperativos

Para la síntesis de supervisores con la técnica de *control por eventos imperativos*, el alfabeto está compuesto por las transiciones mostradas en las Tablas 4.4 y 5.1. La naturaleza de cada transición que corresponde a los componentes elementales se preserva en toda la estructura jerárquica, sin embargo las transiciones informativas que hacen referencia al fin de una tarea (*104, 204, 304, 404, 504, 604*) cambian su naturaleza dependiendo del nivel en la jerarquía (como se explicó en la sección 3.2.2 este cambio de naturaleza no afecta las propiedades de confiabilidad ni realizabilidad). El conjunto de transiciones imperativas está dado por: $\Sigma_{imp} = \{103, 203, 303, 403, 503, 603\}$.

En la Figura 5.1 se muestran los autómatas de entrada para la síntesis del supervisor FDESP. En este nivel las transiciones *104* y *204* son controlables, ya que son las que notifican al nivel superior que se ha finalizado la tarea de despachar. Este supervisor se sintetizó de manera que el comportamiento esté restringido a que se despache de manera secuencial, se agregó esta especificación para tener un supervisor de pocos estados que sea visible, sin embargo se pudo haber diseñado con la capacidad de realizar la tarea en pseudo-paralelo (es decir que visualmente, ambos despachadores saquen materia prima de manera simultánea, se le denomina pseudo-paralelo porque el código es secuencial y un despachador recibirá la orden microsegundos antes que el otro, según como se haya expuesto en la *receta*). Otra alternativa hubiera sido diseñar un par de supervisores, uno para cada despachador.

En la Figura 5.2 se presentan los autómatas de entrada para sintetizar el supervisor FALIM1. La figura incluye los modelos de los componentes elementales, comportamientos causales y especificaciones necesarias para la síntesis del supervisor. El modelo de la planta a lazo abierto resulta de la sincronización de los componentes elementales involucrados y los modelos de los comportamientos causales, indicando que la *fase* no puede terminar antes de que los motores sean detenidos. La operación deseada está descrita por medio de cinco especificaciones funcionales modeladas mediante autómatas. La especificación monolítica se obtiene de la sincronización de las especificaciones individuales.

En este nivel las transiciones imperativas *303* y *305* (alimentar materia prima tipo A y B respectivamente) son entradas, puesto que en este nivel el supervisor está esperando a que el nivel superior le indique cuál de los dos tipos de materia prima se debe alimentar a la estación. Las transiciones informativas correspondientes *304* y *306* son controlables en este nivel, cuando se ha terminado una tarea el supervisor ejecuta la transición correspondiente para notificar al supervisor superior. Estas cuatro transiciones son empleadas como mecanismo de sincronización entre niveles de supervisión correspondientes al proceso en la estación de trabajo número 1.
Siguiendo el mismo procedimiento se obtiene el supervisor de *fase* FME1 (Figura 5.3), en donde la transición informativa *504* es controlable.

La Figura 5.5 muestra el supervisor ODESP, como no hay ninguna especificación extra que cumplir, éste se obtiene únicamente al hacer la proyección natural sobre las transiciones

103, 104,203 y *204* en el supervisor de *fase* FDESP. En este nivel, las transiciones *104* y *204* son incontrolables.

Como ejemplo de síntesis de supervisor de nivel *operación* en la Figura 5.4 se muestran los autómatas de entrada para sintetizar el supervisor OPROC1. En este caso, para obtener los componentes elementales de este nivel de jerarquía se hace la proyección natural sobre las transiciones *303,304,305* y *306* en el supervisor de *fase* FALIM1 y también la proyección natural sobre las transiciones *503* y *504* del supervisor de *fase* FME1. La planta abstracta de este nivel correspondería a la sincronización de ambos componentes elementales resultantes de la proyección. Posteriormente se diseñan las especificaciones de secuencia y otra más para indicar que la capacidad máxima de la celda es de dos piezas de materia prima por estación; es decir, no se podrán manufacturar productos de más de dos piezas, esta sería una especificación de seguridad para no saturar la celda. En este nivel, las transiciones *304,306* y *504* son incontrolables.

Las Figuras 5.6 y 5.7 corresponden a las *recetas* para despachar materia prima (RDESP) y procesar en la estación de trabajo número 1 (RPROC1). Las recetas se modelan como una secuencia ordenada de procesos. En este ejemplo se elaboró una receta para cada procedimiento (despachar materia prima, procesar en estación 1 y procesar en estación 2). En la receta RDESP se establece la secuencia para despachar 4 piezas blancas y 4 color café (alternando colores). En la receta RPROC1 se establece la secuencia para elaborar dos productos formados por dos bloques de materia prima cada uno (primero café y luego blanca), de manera semejante en la receta PROC2 se deben indicar las acciones a ejecutar para manufacturar dos productos de color blanco-café. Las transiciones *104, 204, 304, 306* y *504* son incontrolables en la receta.

La síntesis de los supervisores que corresponden a las actividades de la estación de trabajo número 2 se diseñan de manera semejante, únicamente cambiando a las transiciones correspondientes. La propiedad de confiabilidad se puede comprobar de manera visual en todos los supervisores debido a que los evento imperativos solamente puede ser ejecutados a partir de un estado de decisión (en este caso el estado inicial). Esta propiedad se puede obtener por construcción, partiendo del diseño de los modelos de los componentes elementales y comportamientos causales, tratando de lograr que todas las transiciones *imperativas* salgan únicamente de estados de decisión, lo cual puede lograrse fácilmente en algunos sistemas de manufactura similares al de este ejemplo, donde el inicio y fin de tareas es claro. La confiabilidad es *transitiva*, por lo que los supervisores sintetizados para una planta *confiable*, de cualquier nivel de jerarquía, preservan la propiedad.

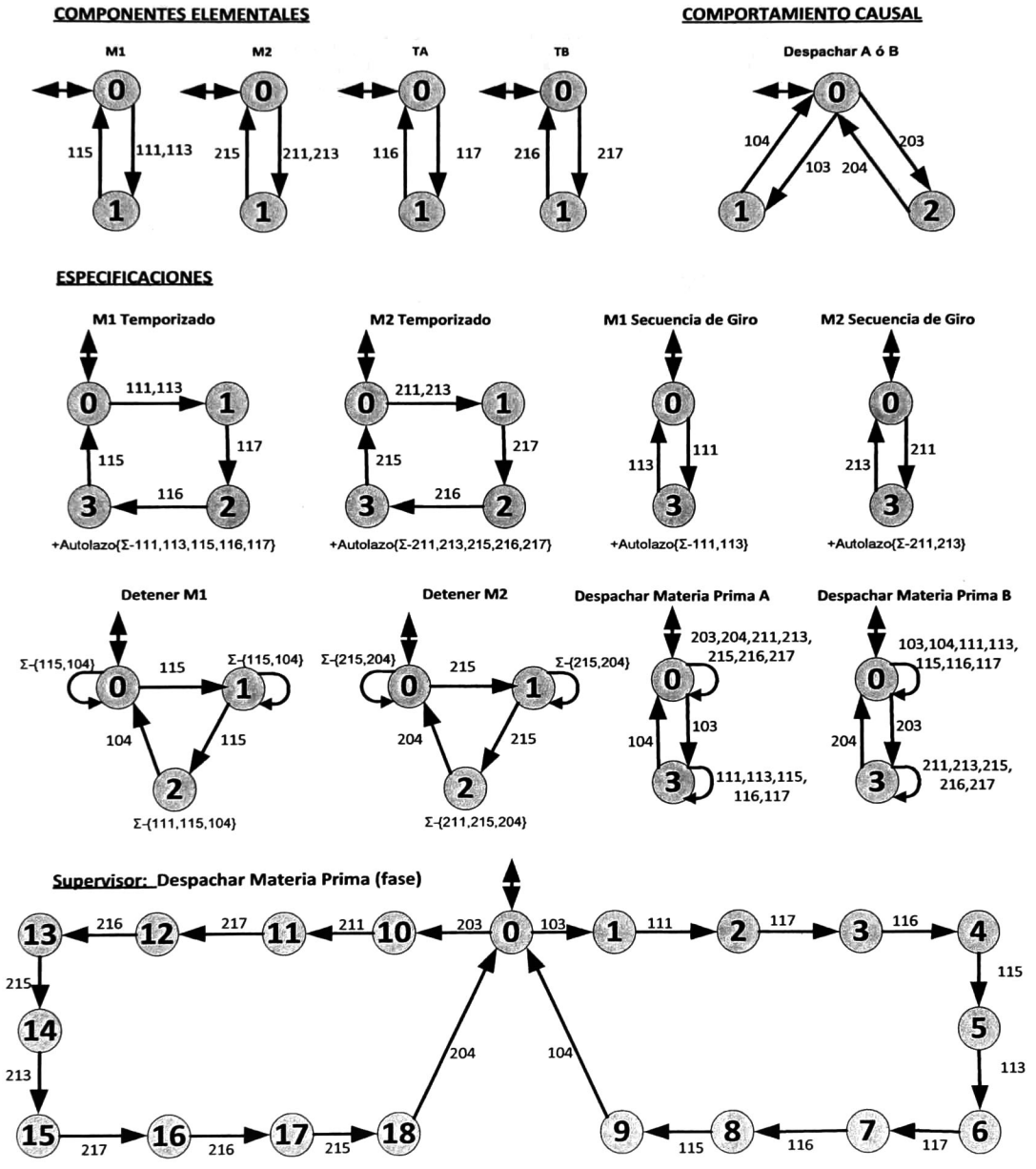


Figura 5.1: Síntesis del Supervisor FDESP1

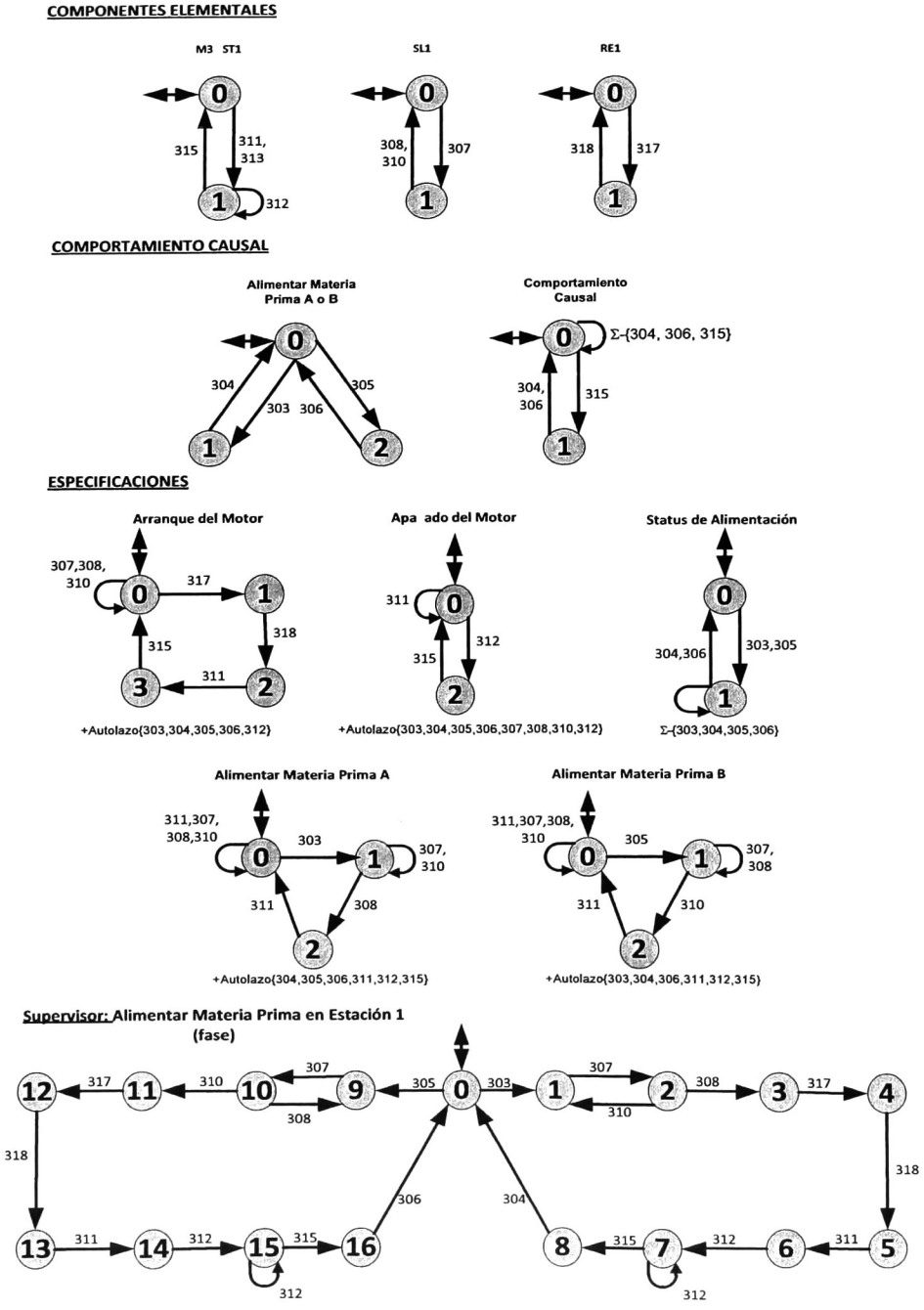


Figura 5.2: Síntesis del Supervisor FALIM1

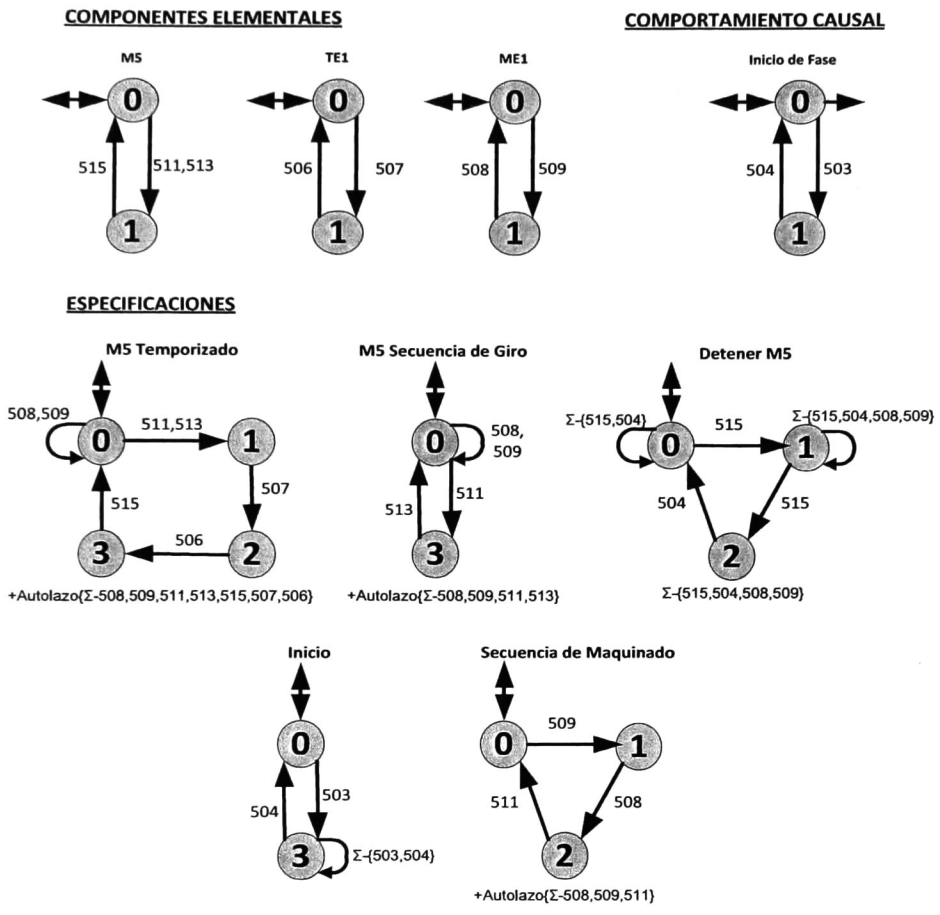


Figura 5.3: Síntesis del Supervisor FME1

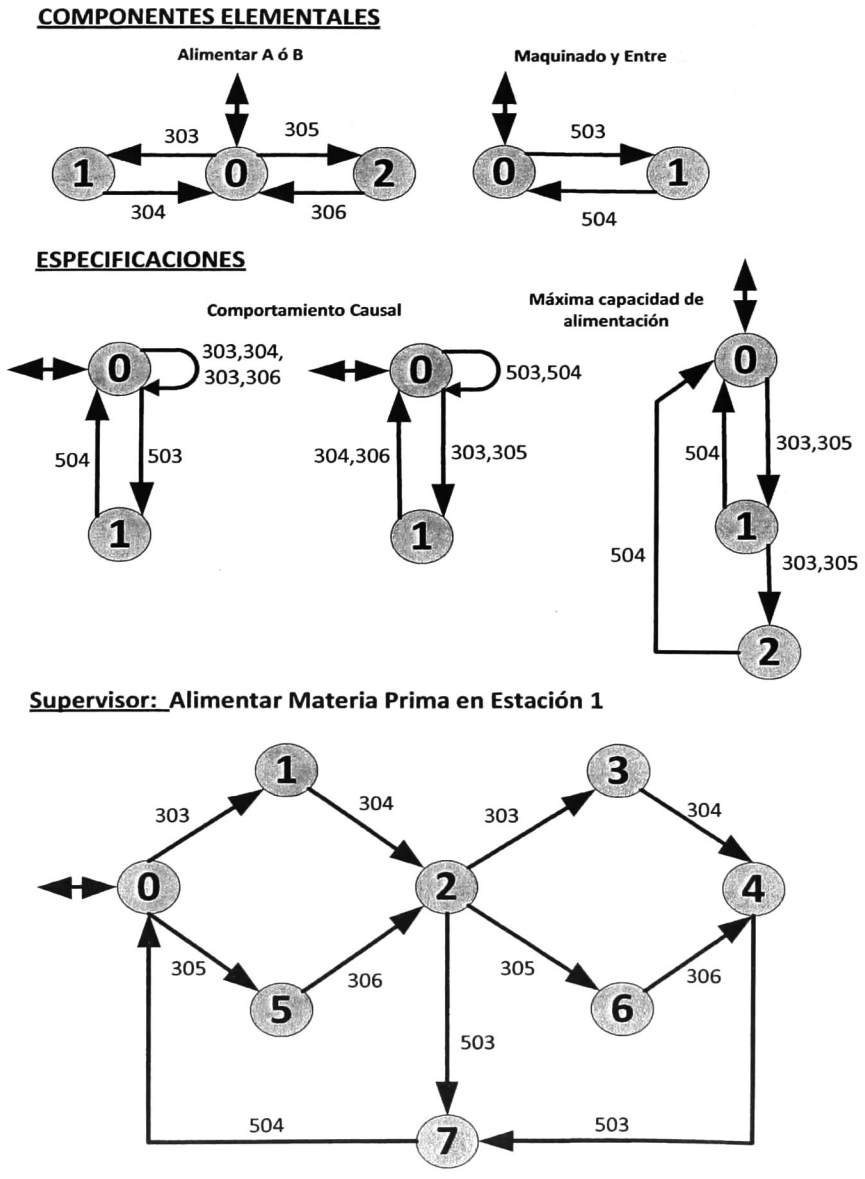


Figura 5.4: Síntesis del Supervisor OPROC1

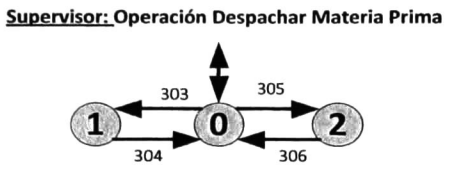


Figura 5.5: Síntesis del Supervisor ODESP

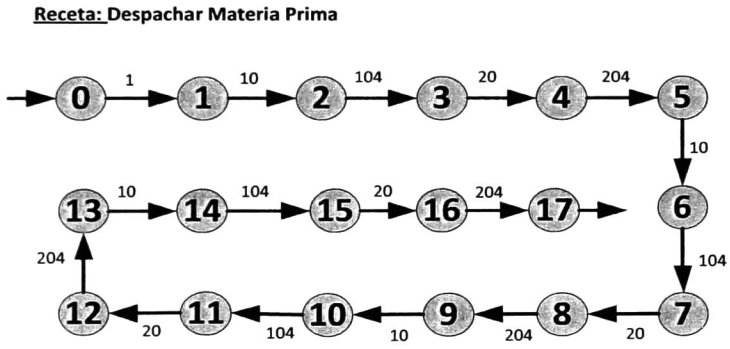


Figura 5.6: Receta RDESP

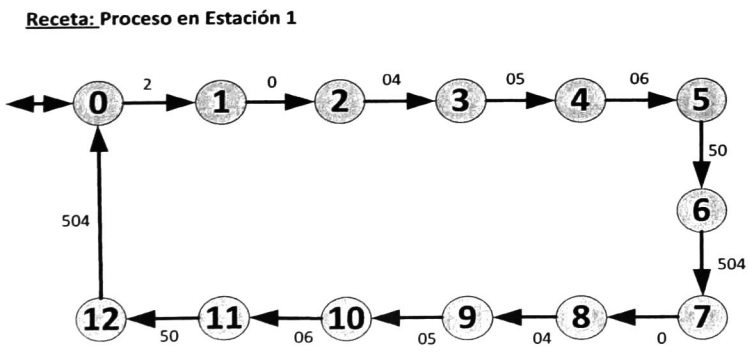


Figura 5.7: Receta RPROC

5.2. Síntesis de Supervisores: Vocalización

En esta sección se presenta la aplicación de la técnica de *vocalización* al prototipo SAM Lego®. La Tabla 5.2 muestra los eventos vocales, para este caso todos resultaron ser controlables. Los cinco supervisores de nivel *fase* se obtuvieron como se explica en la sección 5.1, posteriormente se vocalizaron sus estados para obtener los supervisores de nivel *operación*, vocalizando a su vez éstos se obtuvieron las abstracciones correspondientes, se definieron especificaciones y se sintetizaron los *procedimiento de receta*. El criterio para *vocalizar* determinados estados fue tratar de imponer un inicio y fin de tarea, de manera semejante a con la técnica de *control por eventos imperativos*. Es decir, vocalizar dos estados por cada proceso diferente, para poder establecer la secuencia que corresponda al proceso completo en el nivel superior; teniendo en consideración la presencia de nodos socios. Para este ejemplo, puede ser sencillo verificar que se tenga una estructura SOCC por inspección visual, sin embargo para sistemas complejos se parte de una vocalización y se verifica con el algoritmo *outconsis* de TCT, mediante el cual podemos determinar si hay algún nodo *antecedente* y proceder a cambiar la vocalización de los estados (por lo general la propuesta de vocalización que resulta del algoritmo *outconsis* es más compleja, en el sentido de que se vocalizan más estados de los que son necesarios, se puede obtener una solución con menos estados vocalizados si se re-asignan las salidas).

En la Figura 5.8 se muestran los supervisores sintetizados para despachar materia prima.

A los estados del supervisor FDESP se le asignan las salidas *vocales* (estados 1, 9, 10, 18), logrando que sea consistente en salidas de control mediante el algoritmo *"outconsis"* de TCT (este procedimiento es el que clasifica los eventos vocales en controlables e incontrolables, duplicando a lo mucho el número de símbolos de salida no silenciosos).

Enseguida, mediante el algoritmo *"hiconsis"* se verifica que sea estrictamente consistente en salidas de control (SOCC). Finalmente, una vez que se ha logrado tener una estructura SOCC, mediante el algoritmo *"higen"* se obtiene la abstracción, que en este caso corresponde al supervisor ODESP del nivel *operación*.

El supervisor ODESP se vocaliza a su vez (siguiendo el mismo procedimiento descrito) y se obtiene la abstracción que resulta ser una MEF con un solo estado y con las transiciones *151* y *251* en autolazo en su estado inicial marcado. Esta abstracción se sincroniza con la especificación que corresponde a la secuencia de ejecuciones deseada (despachar materia prima tipo A luego tipo B sucesivamente hasta tener cuatro bloques de cada tipo) y se obtiene el supremo controlable que corresponde a la receta RDESP mostrada.

Por ejemplo, la receta RDESP una vez que inicia, ejecuta la transición *151*, lo que significa que en el supervisor inmediato inferior (ODESP) se deben *habilitar* las transiciones controlables que lleven al estado vocalizado con la transición *151* (es decir en ODESP se habilita la transición *231* para llegar al estado 2 que es el vocal deseado).

A su vez, ejecutar la transición *231* en ODESP implica que este supervisor *habilite* la transición controlable *103* en FDESP.

Es decir, la ejecución de una transición *vocal* en un nivel superior de supervisión implica

Tabla 5.2: Eventos vocales

Vocales	Descripción
131	Vocal de Nivel Fase: Marca el inicio de despachar materia prima tipo A
141	Vocal de Nivel Fase: Marca el fin de despachar materia prima tipo A
231	Vocal de Nivel Fase: Marca el inicio de despachar materia prima tipo B
241	Vocal de Nivel Fase: Marca el fin de despachar materia prima tipo B
151	Vocal de Nivel Operación: Iniciar a despachar materia prima tipo A
251	Vocal de Nivel Operación: Iniciar a despachar materia prima tipo A
331	Vocal de Nivel Fase: Se detecta materia prima tipo A en estación 1
341	Vocal de Nivel Fase: Se ha alimentado materia prima tipo A en estación 1
351	Vocal de Nivel Fase: Se detecta materia prima tipo B en estación 1
361	Vocal de Nivel Fase: Se ha alimentado materia prima tipo B en estación 1
531	Vocal Nivel Fase: Se inicia el maquinado y entrega de producto en estación 1
541	Vocal Nivel Fase: Se termina el maquinado y entrega de producto en estación 1
551	Vocal Nivel Operación: Se alimenta materia prima tipo A en estación 1
561	Vocal Nivel Operación: Se alimenta materia prima tipo B en estación 1
571	Vocal Nivel Operación: Se entrega producto en estación 1
431	Vocal de Nivel Fase: Se detecta materia prima tipo A en estación 2
441	Vocal de Nivel Fase: Se ha alimentado materia prima tipo A en estación 2
451	Vocal de Nivel Fase: Se detecta materia prima tipo B en estación 2
461	Vocal de Nivel Fase: Se ha alimentado materia prima tipo B en estación 2
631	Vocal Nivel Fase: Se inicia el maquinado y entrega de producto en estación 2
641	Vocal Nivel Fase: Se termina el maquinado y entrega de producto en estación 2
651	Vocal Nivel Operación: Se alimenta materia prima tipo A en estación 2
661	Vocal Nivel Operación: Se alimenta materia prima tipo B en estación 2
671	Vocal Nivel Operación: Se entrega producto en estación 2

que por el canal de comunicación se "envíe" una secuencia de transiciones controlables a habilitar en el supervisor inmediato inferior. Si se trata de una estructura SOCC esto garantiza que al enviar dicha secuencia de transiciones controlables, el "camino" habilitado en el nivel inferior llevará únicamente al estado vocal deseado (sin pasar por ningún otro estado vocal). Dicho con otras palabras, existe una garantía de que lo estipulado en el supervisor superior podrá realizarse en el nivel inferior, sin que pase un evento indeseado antes. De esta manera, todas las transiciones controlables necesitan ser previamente habilitadas antes de poder ser ejecutadas en un estado. Solamente en el nivel de mayor jerarquía, que corresponde a la receta, las transiciones controlables implican ejecuciones porque en este nivel es donde se lleva a cabo la decisión y el control de las tareas.

En la Figura 5.10 se muestra el supervisor de nivel *fase* para alimentar materia prima en la estación de trabajo número 1 (FALIM1).

Como primer intento se vocalizaron los estados 1, 8, 9 Y 16 en el supervisor FALIM1 que son los que corresponden al inicio y fin de la tarea de alimentar materia prima tipo A y B respectivamente, sin embargo al vocalizar estos estados no se obtiene una estructura SOCC. Se puede verificar a simple vista que no es SOCC por la presencia de la transición incontrolable *310* que regresa al estado vocal 1. Por ejemplo, si el estado 1 se vocalizara con la transición α y el estado 8 con la transición β, en la abstracción podría establecer la secuencia α-β. Lo que significa que primero para que ocurra α se debe habilitar la transición *303* y luego para que ocurra β se enviaría la secuencia de habilitación "*307-317-311-315*" Pero al habilitar esta secuencia en FALIM1 se observa que no necesariamente ocurrirá β ya que al habilitar *307* puede ocurrir α nuevamente antes de β (si se da la transición incontrolable *310* en el estado 2), por lo tanto no es SOCC. Si fuera SOCC estaría garantizado que en el supervisor inferior se llegaría al estado vocal deseado sin que antes ocurriera otro evento vocal indeseado. Se solucionó el problema vocalizando los estados 3, 8, 11 y 16.

La Figura 5.11 muestra la vocalización del supervisor FME1 y en la Figura 5.12 se presentan las abstracciones obtenidas a partir de los supervisores FALIM1 y FME1, las cuales se sincronizan junto con las especificaciones para sintetizar el supervisor OPROC1 mostrado en la misma figura.

Finalmente, de la misma manera en que se obtuvo la receta RDESP, para obtener RPROC1 el supervisor OPROC1 se vocaliza a su vez y se obtiene la abstracción que resulta ser una MEF con un solo estado y con las transiciones *551, 561* y *571* en autolazo en su estado inicial marcado. Esta abstracción se sincroniza con la especificación que corresponde a la secuencia de ejecuciones deseada (alimentar, maquinar y entregar producto en la estación de trabajo 1) y se obtiene el supremo controlable que corresponde a la receta RPROC1 mostrada en la Figura 5.9.

El mismo procedimiento de diseño se lleva a cabo para el proceso en la estación de trabajo número 2, solamente haciendo el cambio correspondiente en las transiciones y el cambiando el orden de alimentación en la *receta* RPROC2.

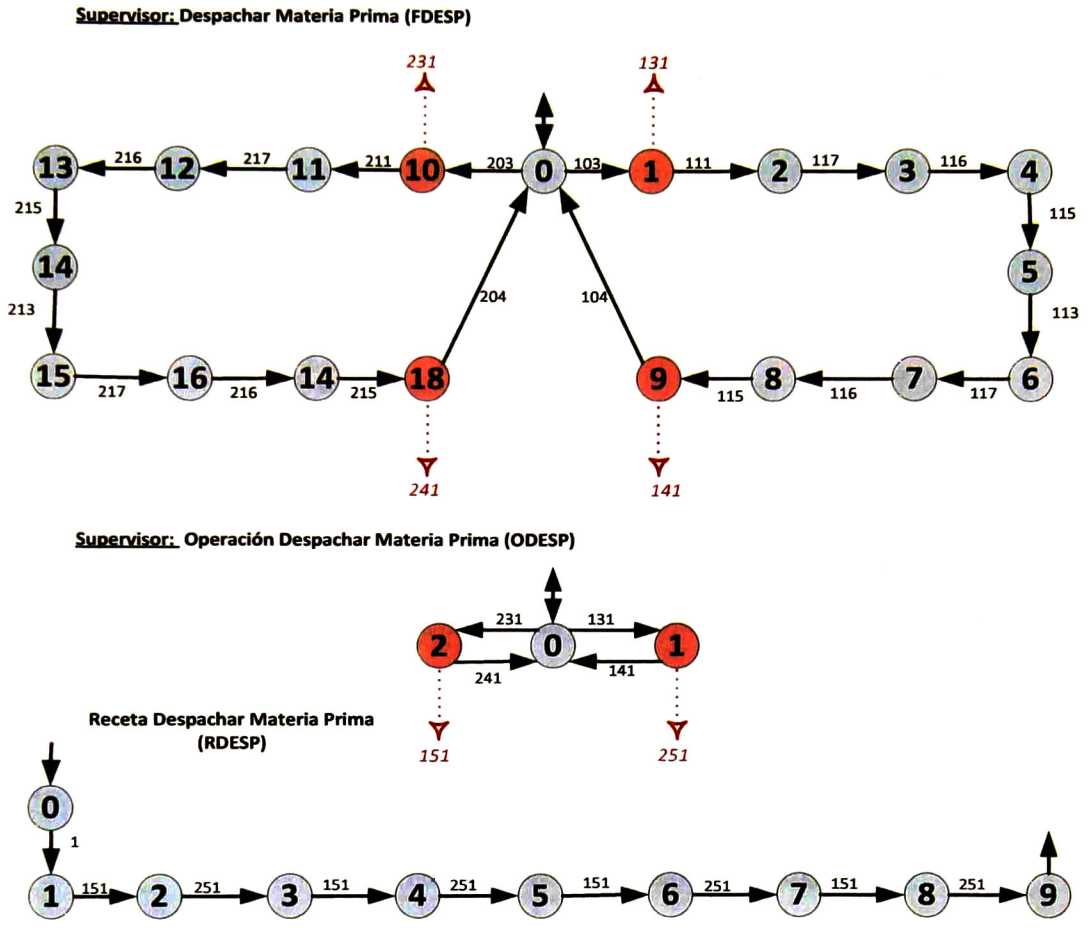


Figura 5.8: Supervisores Vocalizados para Despachar Materia Prima

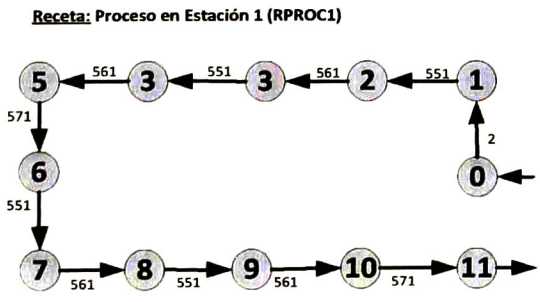


Figura 5.9: Receta para procesar en la Estación 1 (RPROC1)

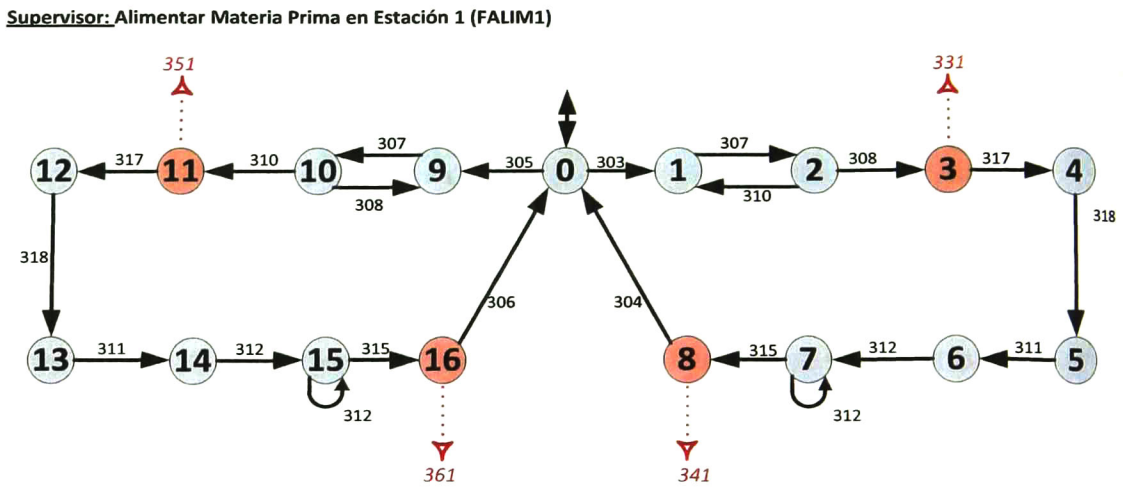


Figura 5.10: Supervisor vocalizado para Alimentar Materia Prima en la Estación 1 (FALIM1)

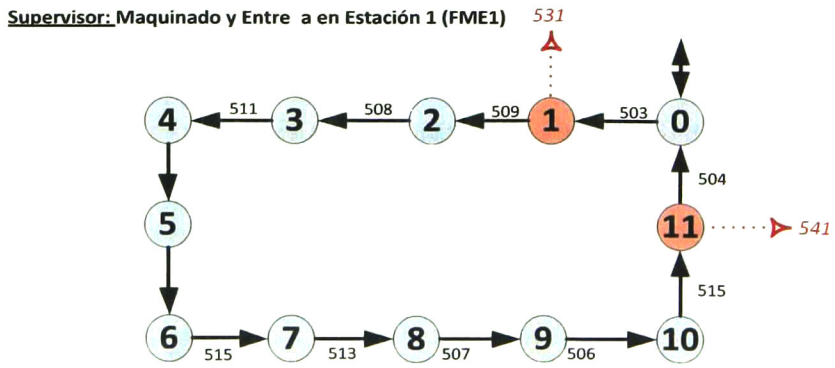


Figura 5.11: Supervisor vocalizado para el proceso de Maquinado y Entrega en la Estación 1 (FME1)

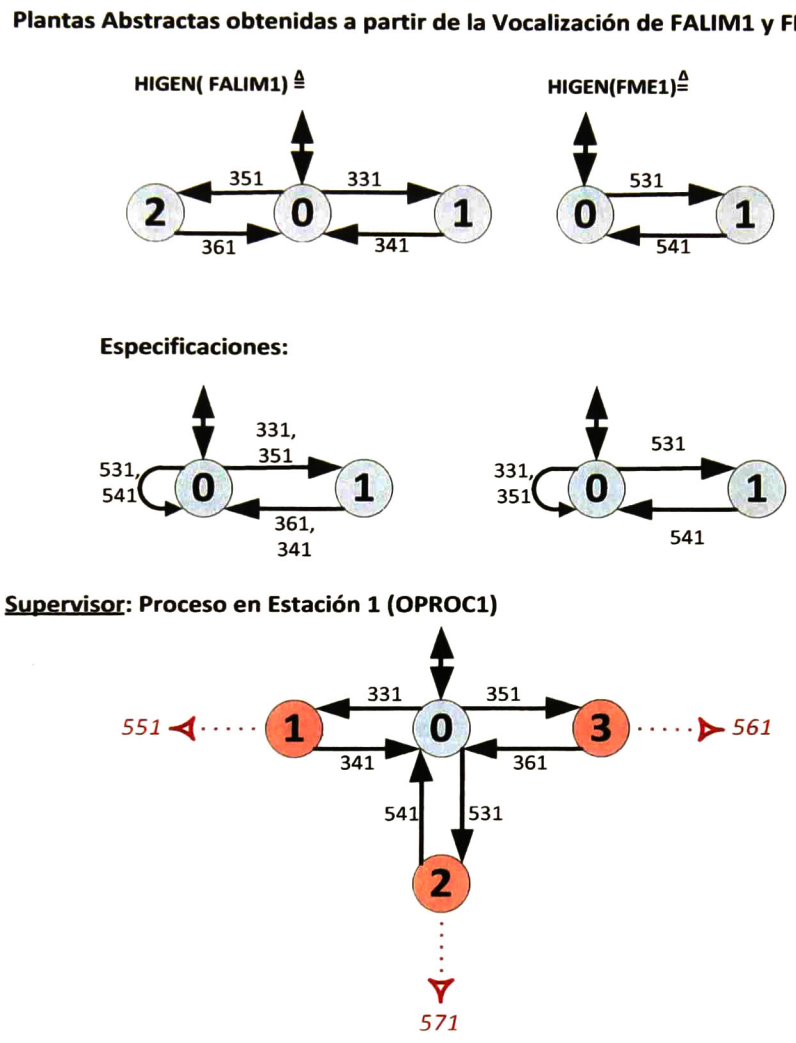


Figura 5.12: Supervisor de operación para el proceso en la Estación 1 (OPROC1)

Capítulo 6

Análisis Comparativo de las Técnicas

En esta sección se presenta una comparativa entre las técnicas de *vocalización* y *control por eventos imperativos*, presentando sus ventajas e inconvenientes tanto en la síntesis como en la implementación, en aplicaciones de manufactura.

6.1. Parámetros de Comparación

Ambas técnicas producen estructuras jerárquicas consistentes en el sentido de que no hay bloqueo entre los niveles de supervisión y no se ve afectada la propiedad de controlabilidad. También, las dos técnicas están bien fundamentadas matemáticamente y cuentan con una metodología precisa para el diseño de supervisores de niveles superiores.

Cuando el alfabeto no es disjunto en los módulos horizontales de supervisión, con ambas técnicas es necesario la verificación del conflicto y en caso de existir se debe coordinar el uso de recursos en una capa superior. Sin embargo, la forma en que logran esta consistencia es diferente, como por ejemplo las condiciones o restricciones en los supervisores, el modelo del canal de comunicación entre niveles y los recursos empleados en su implementación.

Los ámbitos en que se hace la comparación de ambas técnicas son:

- Restricciones o Condiciones en el Modelado
- Flexibilidad de la Arquitectura Jerárquica
- Implicaciones en la implementación y uso de recursos

6.1.1. Restricciones o Condiciones en el Modelado

La técnica de *control por eventos imperativos* exige que los autómatas que describen el modelo del sistema (o los supervisores sintetizados, ya que la propiedad es transitiva) sean *confiables*, esta propiedad puede ser claramente garantizada si se construyen autómatas en donde las transiciones imperativas únicamente estén habilitadas en *estados de decisión*, esta condición puede ser muy restrictiva sin embargo para sistemas de manufactura es muy

común que no sea difícil de cumplir, ya que por lo general no hay procesos o tareas que se inicien de manera incontrolable. Además, la condición de *confiabilidad* puede relajarse, solamente se necesita cumplir que exista un estado de decisión lograble a partir de $\delta(q, \sigma)$ donde σ es una transición imperativa.

La parte de *consistencia jerárquica* se logra haciendo la verificación de que la *receta* sea *realizable* con respecto a los supervisores involucrados en el nivel inmediato inferior, esto se verifica al comprobar el isomorfismo $Proj(L(M) \parallel L(G))$ con $L(M)$, lo cual es fácilmente logrado con TCT o Supremica. De esta manera se está garantizando que lo estipulado en la *receta* podrá llevarse a cabo paso a paso en el sistema sin que éste se bloquee o el sistema evolucione diferente, en este sentido esta definición es análoga a la condición de estructura *SOCC* definida en *vocalización*.

En el caso de la *vocalización*, la estructura *SOCC* se logra *vocalizando* los estados de manera que no haya *nodos socios*, esta condición es similar a la *confiabilidad* en *control por eventos imperativos* (en el sentido de que ambas propiedades garantizan el no bloqueo entre niveles y la consistencia jerárquica), sin embargo en ésta última técnica se puede lograr dicha condición por construcción y en *vocalización*, aunque se puede identificar la presencia de *nodos socios* a simple vista en modelos de pocos estados, en MEF de numerosos estados será difícil su comprobación visual y tendrá que verificarse con herramientas computacionales.

Además, la solución de estructura *SOCC* que entrega como resultado el algoritmo *hiconsis* de TCT no es necesariamente la mejor, ya que este algoritmo solamente identifica los *nodos socios* y *vocaliza* los *nodos antecedentes*, resultando un alfabeto de símbolos no silenciosos más numeroso, pudiendo haber obtenido la estructura *SOCC* solamente con cambiar los estados vocalizados; es decir, la obtención de una estructura *SOCC* óptima en el sentido de que tenga menor cantidad de estados vocalizados se logra solamente a prueba y error cambiando la asignación de salidas de los estados.

Otro inconveniente en el caso de la *vocalización* en aplicaciones de manufactura, es que por lo general es necesario incluir transiciones *virtuales* de inicio y fin de tarea, creciendo de manera considerable el alfabeto. Por ejemplo, considerando las transiciones definidas en las Tablas 4.4 y 5.1 y el supervisor de *fase* para despachar materia prima (FDESP, Figura 5.8). Si en su síntesis no se hubieran incluido las transiciones de inicio y fin de *fase* (*103,104, 203* y *204*) y se hubieran vocalizado, entre otros, los estados en donde salen las transiciones *115* y *215*; entonces el supervisor del nivel superior no podría hacer que se complete satisfactoriamente la tarea de despachar, ya que se dejaría un motor encendido. Esto pasaría porque, al no poder vocalizar el nodo inicial, el supervisor del nivel superior enviaría la cadena correspondiente para llegar hasta el estado donde está habilitada la transición *115* ó *215* dejando el motor sin apagar (ya que llega hasta ese estado pero no se ejecuta la transición *115* ó *215* que corresponden al apagado del motor) y si se enviara otra cadena para llegar al siguiente estado vocal, al no poder ser el estado inicial, entonces se vería obligado a dejar encendido el motor en el sentido opuesto (porque se habrían ejecutado las transiciones *111* ó *211* que son las que saldrían del estado inicial).

De esta manera, en la técnica de *vocalización*, el alfabeto global puede ser muy grande, mayor al empleado en la técnica de *control por eventos imperativos*, ya que además de tener

que incluir las transiciones de inicio y fin de tarea, el alfabeto de cada nivel es disjunto (cada abstracción está formada por los eventos vocales), es decir se tiene que crecer el alfabeto por cada nivel de jerarquía. Si por ejemplo se *vocalizaran* todos los estados del nivel *fase* el alfabeto crecería tanto como estados tuviera dicho supervisor de *fase*.

Entonces se concluye que en cuanto a restricciones y condiciones en el modelado, puede ser más compleja y restrictiva la técnica de *vocalización*.

6.1.2. Flexibilidad de la Arquitectura Jerárquica

Como se ha mencionado, uno de los propósitos de la construcción de estructuras jerárquicas es lograr arquitecturas flexibles, en el sentido de que modificaciones en el plan de producción o alteraciones en el sistema impliquen en la menor medida posible cambios en el diseño del sistema de control (re-sintetizar supervisores).

Ambas técnicas son igualmente flexibles en cuanto al *cambio en las reglas de producción*, esto es así por la forma en que se propone el modelo de control de procedimientos siguiendo el estándar ISA-88, colocando en el mayor nivel de jerarquía la *receta* y solamente será necesario modificar esta receta, independientemente de la técnica empleada. La flexibilidad dependerá en gran medida del diseño del modelo de control de procedimientos, partiendo de que es igual para ambas técnicas, esta propiedad será muy semejante.

Sin embargo, cuando hay cambios en la planta y por consiguiente cambia el procedimiento de *fase* en alguna medida, entonces tener una estructura obtenida mediante la técnica de *control por eventos imperativos* podría resultar más flexible (o igual que en *vocalización*, pero nunca menos flexible).

Esto se debe por la metodología para obtener los supervisores superiores. En ambas técnicas se tendría que re-sintetizar el supervisor de *fase* relacionado; sin embargo, si no se alteran las transiciones imperativas e informativas, la proyección natural sobre este supervisor no se verá afectada y el supervisor de nivel *operación* quedaría intacto en el caso de *control por eventos imperativos*. Es decir solamente se le agregarían "pasos" al procedimiento de *fase*, sin afectar el nivel superior. Sin embargo, en *vocalización* al ser una abstracción que se construye de manera incremental probablemente se tendrá que re-sintetizar también el supervisor superior.

6.1.3. Implicaciones en la implementación y uso de recursos

En la sección 3 se expuso el esquema de traducción según la técnica empleada. En el caso de *vocalización* se requieren mucho más recursos:

- Por cada transición controlable se tienen dos variables globales (una que es habilitada por el supervisor superior y la otra que indica la ejecución de la transición controlable). En *control por eventos imperativos* solamente es una variable por transición controlable o incontrolable.

- Al ser el alfabeto empleado en *vocalización* mayor al de *control por eventos imperativos* implica que se tienen que definir más variables globales en total. Por lo general será mayor el alfabeto de una arquitectura jerárquica obtenida con *vocalización*, ya que cada nivel de abstracción crece su alfabeto al definir eventos *vocales* y en ocasiones también se tienen que definir las transiciones de inicio y fin de fase en aplicaciones de manufactura.

- La MEF que modela el supervisor en *vocalización* no contiene de manera explícita mucha información relacionada con los canales de comunicación y que es necesaria para su implementación:

  - Nivel de jerarquía en que se encuentra el supervisor que se desea implementar.
  - Conjunto de cadenas de transiciones controlables a habilitar dado cada evento vocal (las cuales se obtienen mediante un algoritmo de TCT en la etapa del diseño).

  En contraparte, en la técnica de *control por eventos imperativos* solamente se requiere definir las transiciones imperativas como información extra además de la MEF que modela al supervisor.

En el caso de *control por eventos imperativos* se emplean menos recursos del PLC, al tener que definir menos variables globales y no tener que incluir la información de los canales de comunicación como en el caso de la *vocalización*.

Aunque ambos algoritmos se pueden programar para obtener traducciones automáticas, puede resultar más complicado el caso de la *vocalización* por la cantidad de información extra necesaria para lograr una traducción adecuada.

Se realizó una versión beta de una herramienta de traducción automática que recibe como entrada la MEF que modela al supervisor obtenido mediante la técnica de *control por eventos imperativos* y la declaración de las transiciones imperativas y entrega como resultado su traducción en texto estructurado. No se desarrolló herramienta para el caso de supervisores vocalizados por su complejidad.

Capítulo 7

Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones finales de la tesis y el trabajo futuro relacionado con la misma.

7.1. Conclusiones

Todas las técnicas de control supervisor jerárquico-modular ayudan a evitar el problema de la explosión del espacio de estados reduciendo la complejidad asociada con la síntesis de supervisores y ahorrando recursos computacionales, además de producir arquitecturas más flexibles y reconfigurables. Tanto la *vocalización* como la técnica de *Control por eventos imperativos* producen estructuras jerárquicas consistentes en el sentido de que no hay bloqueo entre los niveles de supervisión (la primera lo logra mediante la propiedad SOCC y la segunda mediante la propiedad de confiabilidad).

Aunque ambas técnicas cumplen con el mismo propósito y logran resultados equivalentes, cada metodología tiene sus ventajas y diferencias en la síntesis de supervisores y su implementación, resultando ser la técnica de *Control por eventos imperativos* la que reúne mayores ventajas en aplicaciones de manufactura.

Se puede concluir, al menos para este ejemplo (y posiblemente para aplicaciones de manufactura similares, en donde los procesos estén claramente delimitados y los inicios y fines de tarea puedan ser definidos), que en cuanto a *restricciones y condiciones en el modelado*, es más compleja y restrictiva la técnica de *vocalización* por las siguientes razones:

- La condición de confiabilidad en *control por eventos imperativos* se logra fácilmente por construcción mientras que la propiedad análoga de *estructura SOCC* (no presencia de nodos socios) se logra a prueba y error mediante herramientas computacionales.
- El alfabeto empleado en la técnica de *vocalización* puede ser mayor que el de *control por eventos imperativos*, ya que en aplicaciones de manufactura, además de que cada nivel de supervisión tiene su alfabeto completamente disjunto (crece por cada nivel

de abstracción), se deberán incluir en muchas ocasiones transiciones de inicio y fin de *fase*. En *control por eventos imperativos* solamente es necesario la inclusión de éstas últimas.

En cuanto a *flexibilidad* de la estructura de control se concluye que ambas técnicas son prácticamente igual de flexibles ante cambios en el sistema como modificaciones en el plan de producción. La flexibilidad dependerá en gran medida del diseño del modelo de control de procedimientos.

En cuanto a *facilidad en la implementación y uso de recursos* se concluye que es mejor la técnica de *control por eventos imperativos*, ya que el algoritmo empleado en la traducción no requiere de la información de los canales de comunicación presentes en *vocalización* y emplea menos recursos en el PLC. Puede ser más sencilla la automatización de la implementación de supervisores diseñados con esta técnica debido a que se requiere menos información: solamente la MEF que modela al supervisor y la definición de transiciones imperativas.

En contraparte, para automatizar la traducción de supervisores obtenidos mediante *vocalización* se requiere conocer, además de la MEF que lo modela: el nivel de jerarquía al que pertenece y la cadena de transiciones controlables que se debe habilitar en el supervisor inferior dado un evento vocal (esta cadena de transiciones se debe obtener para cada vocal controlable y se requiere de un algoritmo para obtenerlas).

7.2. Trabajo Futuro

En cuanto al prototipo y su integración con el equipo de control se podrían hacer algunos ajustes de carácter mecánico, por ejemplo:

- Mejorar el sistema de bandas, colocando nuevos motores para que gire más rápido, alinear el sistema de engranaje y nivelar las bandas para evitar que la materia prima se gire al cambiar de banda.
- Mejorar el sistema de alimentación de materia prima en la estación de trabajo.
- Aumentar la complejidad del sistema introduciendo nuevos componentes.

En este trabajo se desarrolló una primera versión de herramienta de traducción automática entre supervisores modelados en MEF a texto estructurado, como trabajo futuro se podrían realizar herramientas más robustas y completas para la implementación de supervisores (por ejemplo una aplicación que conjugue varias herramientas para automatizar en la mayor medida la implementación, verificando propiedades como realizabilidad y permitiendo elaborar la receta de manera visual y capaz de comunicarse con el PLC para descargar directamente los programas).

Se implementaron los supervisores como dispositivos de control, como trabajo futuro también se podrían adecuar estos esquemas de traducción de manera que cumplan estrictamente con la definición de supervisores, es decir, que no fuercen la ejecución de transiciones sino que se tengan por separado el controlador del proceso y su supervisor.

En este trabajo no se considera la noción de un observador de estados, se supuso todo el estado disponible (el prototipo mide todas las señales incontrolables), esto puede restringir el tipo de problemas que se puedan resolver. Existen escenarios reales en los que la complejidad del sistema complica medir directamente todos sus estados. En otras situaciones, la falla de uno o más de sus sensores provoca que la información del estado esté incompleta para el controlador. Estos son solo algunos casos de sistemas que se requiere controlar bajo la observación parcial de su estado. Se podría complementar este trabajo analizando el caso de la observación parcial del estado o la capacidad de detección de fallas y determinar si los esquemas de traducción propuestos necesitan ser ampliados.

Adicionalmente, el uso de autómatas como herramienta de modelado de SED puede resultar impráctico o poco eficiente; se pueden representar SED de manera mucho más compacta con otras herramientas como las redes de Petri. Para la síntesis de los supervisores no resulta problemático el hecho de que los modelos del sistema o de la especificación sean muy grandes al modelarlos con autómatas, gracias tanto al avance de las herramientas computacionales como a técnicas de modularización; sin embargo sería interesante explorar otras técnicas de síntesis de supervisores y de estructuras jerárquicas empleando otras herramientas de modelado y compararlas con las presentadas.

Apéndice A

Esquemáticos de la Interfaz Electrónica

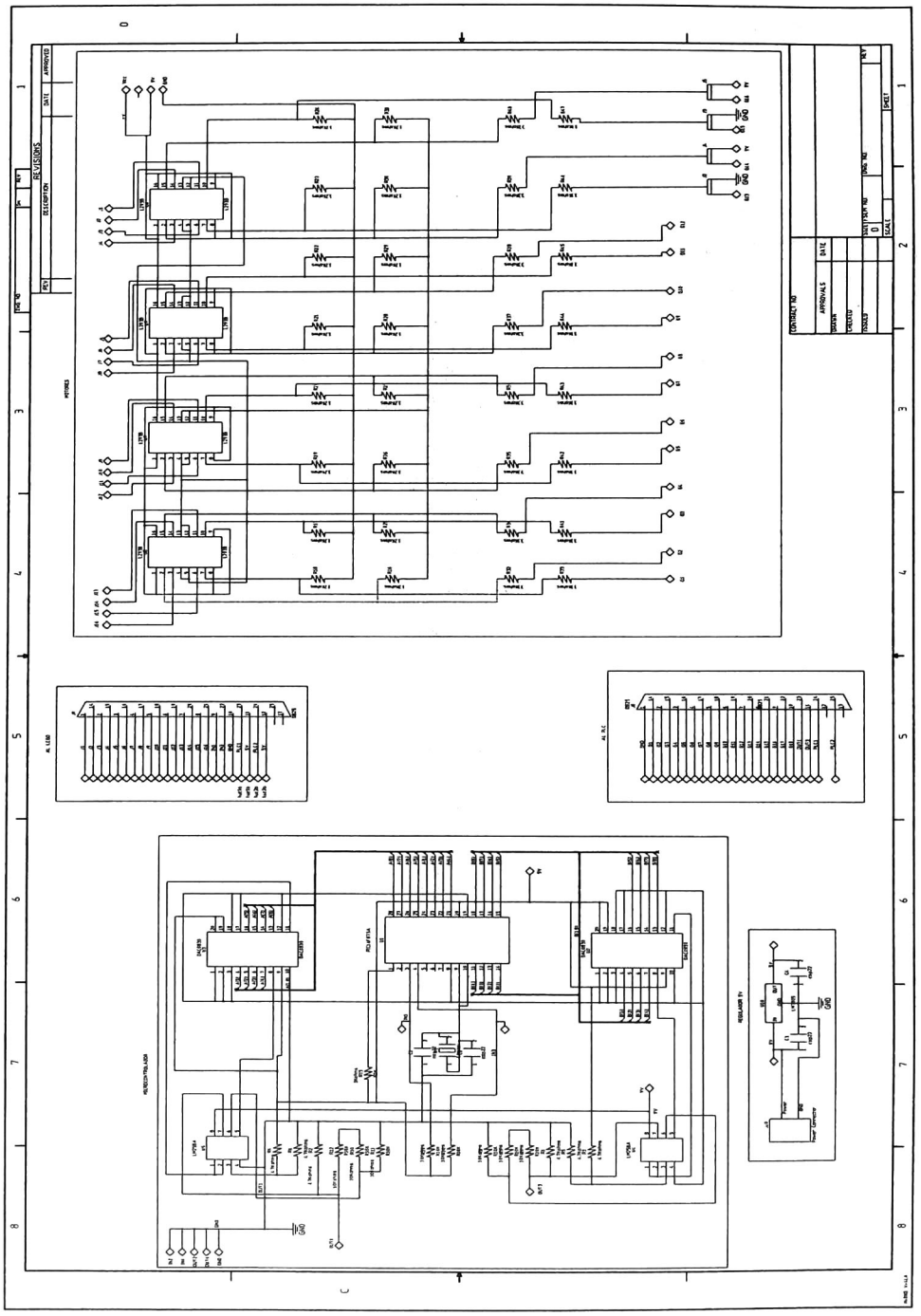


Figura A.1: Esquemático de la tarjeta

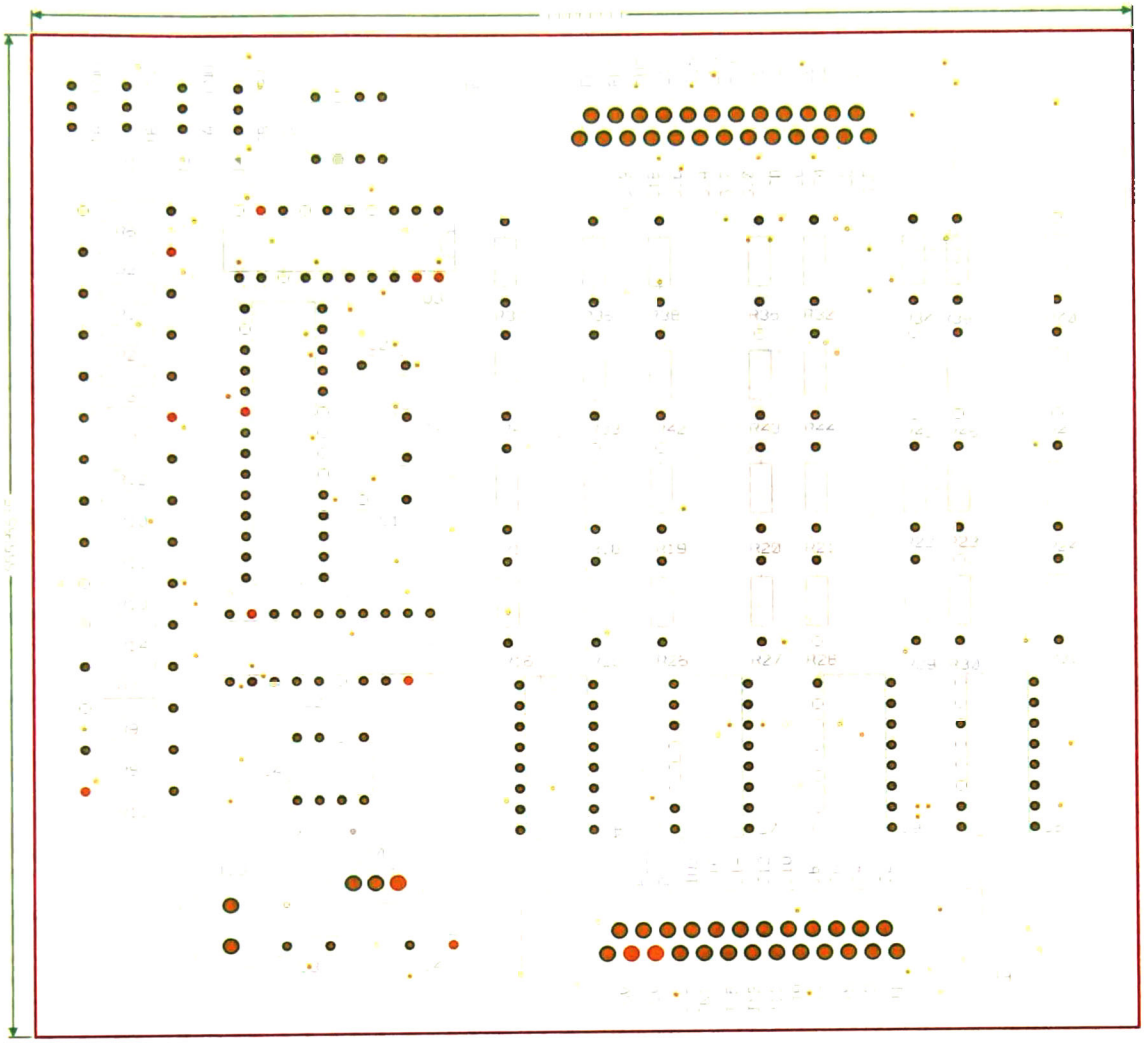


Figura A.2: Colocación de componentes en la tarjeta

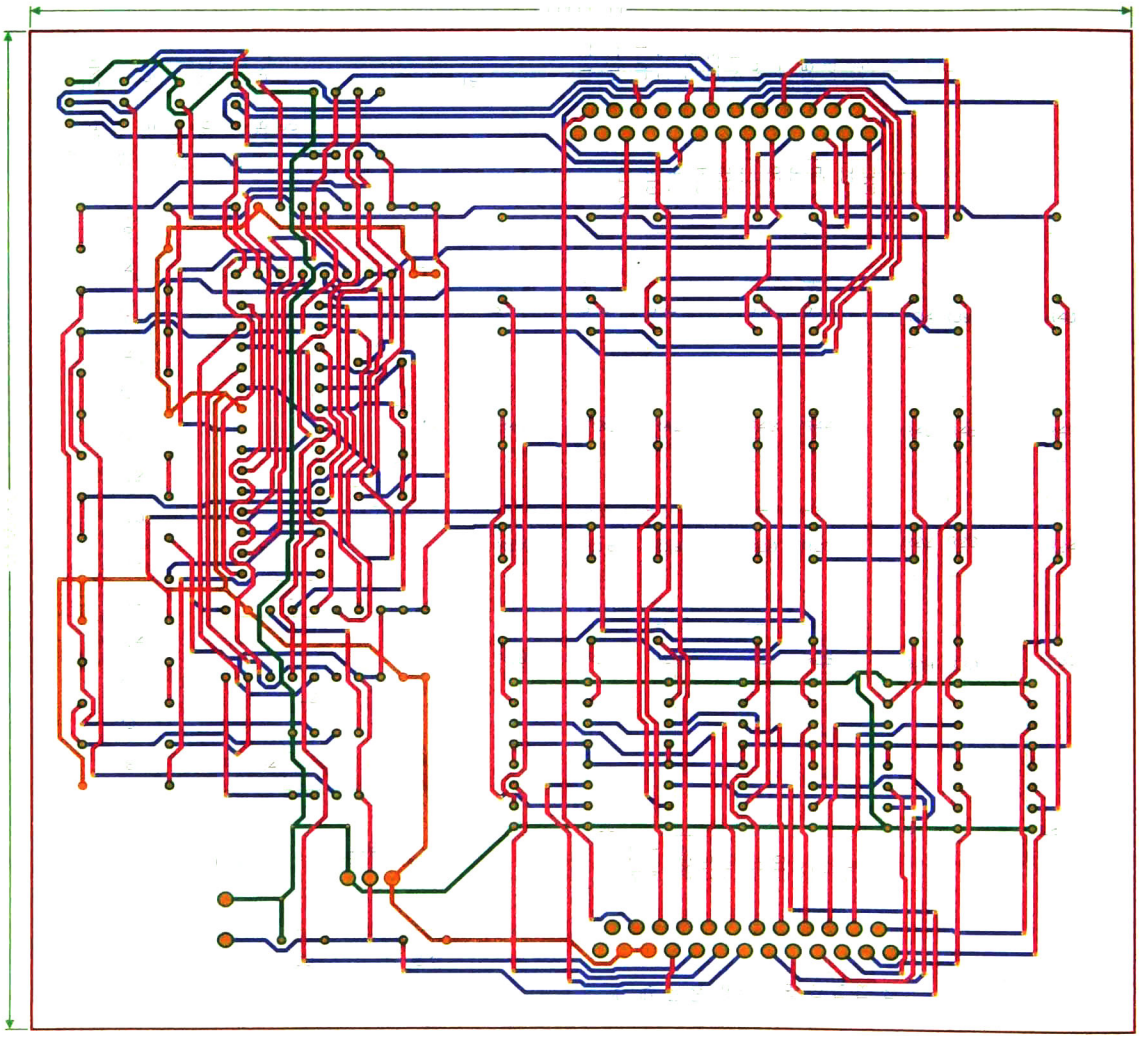


Figura A.3: Ruteo de la tarjeta

Apéndice B

Ejemplo de implementación en LD de un supervisor vocalizado

Las Figuras B.1 y B.2 muestran dos supervisores sintetizados con la técnica de vocalización, uno para el nivel superior (HCGHI) y otro que corresponde al nivel inferior (HCGLO), para el control de una línea de transferencia. Estos supervisores fueron tomados de la sección 5.6 de [11].

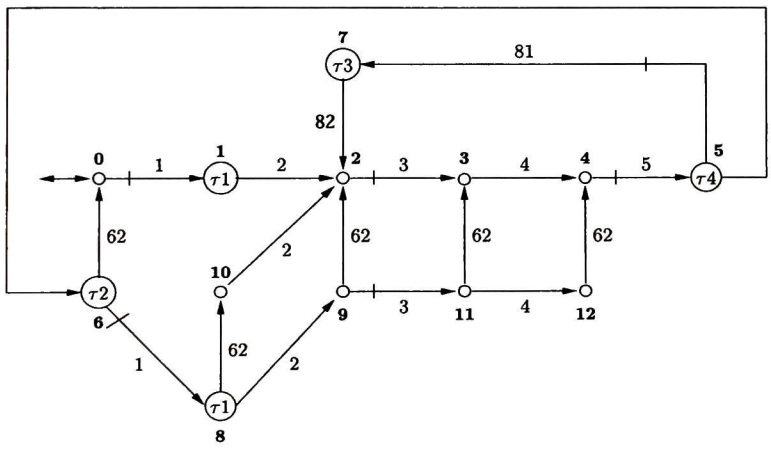


Figura B.1: Supervisor del nivel inferior (HCGLO)

En la Figura B.3 se muestra la implementación en LD que corresponde a los primeros seis estados del supervisor del nivel bajo (HCGLO), utilizando las reglas propuestas en la sección 3.3.2. No se incluyeron todos los estados debido a que los seis primeros bastan para mostrar todos los casos y aplicar todas las reglas definidas en dicha sección.

En las Figuras B.4 y B.5 se observa la implementación en LD del supervisor del nivel superior (HCGHI).

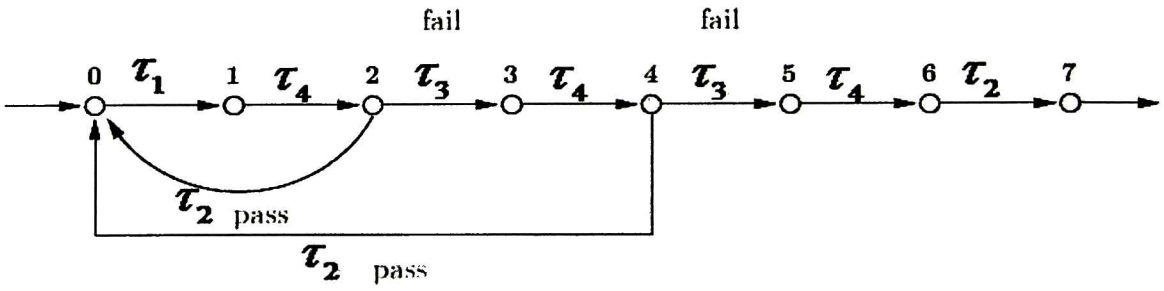


Figura B.2: Supervisor del nivel superior (HCGHI)

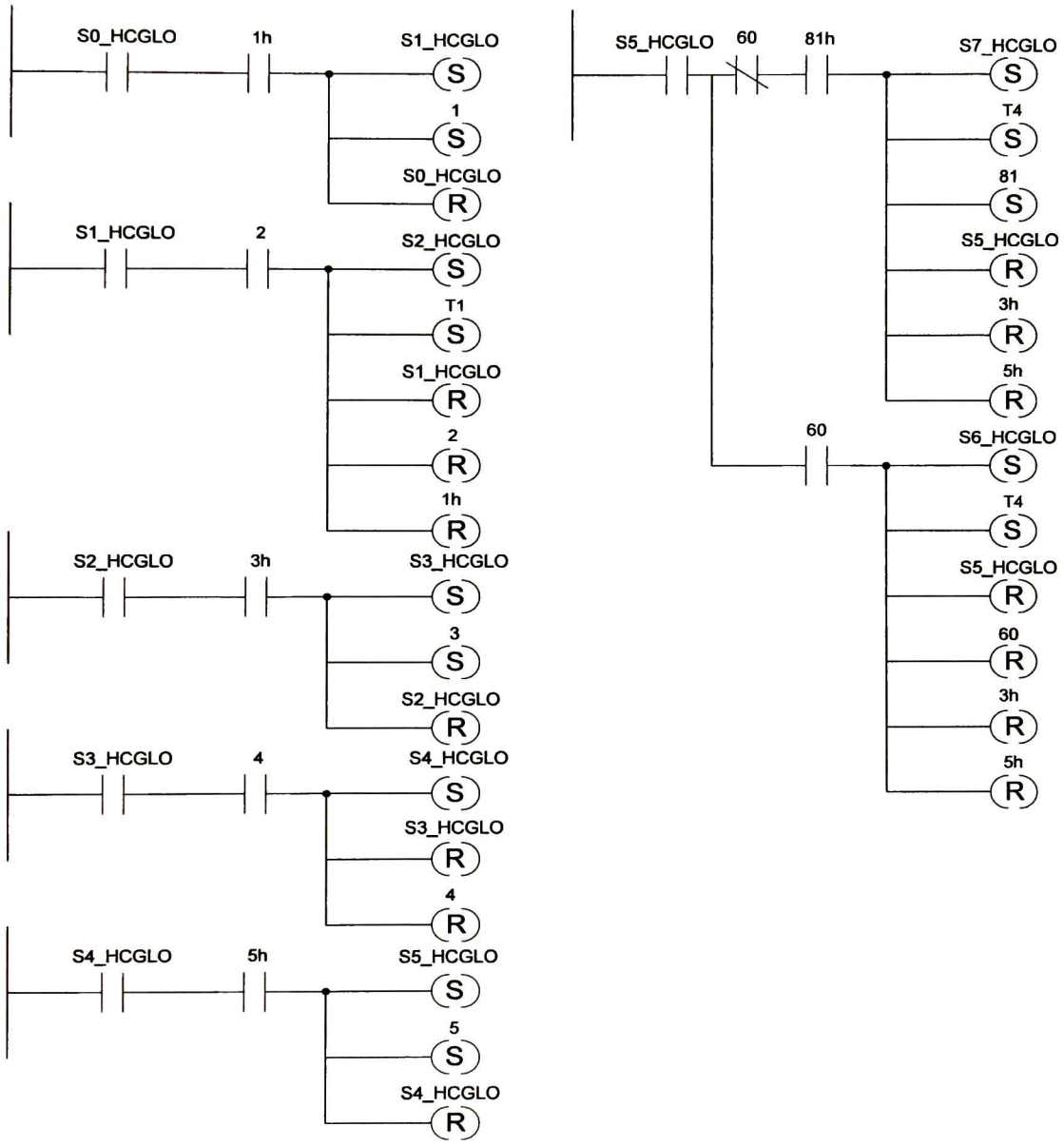


Figura B.3: Implementación en LD del Supervisor del nivel inferior (HCGLO)

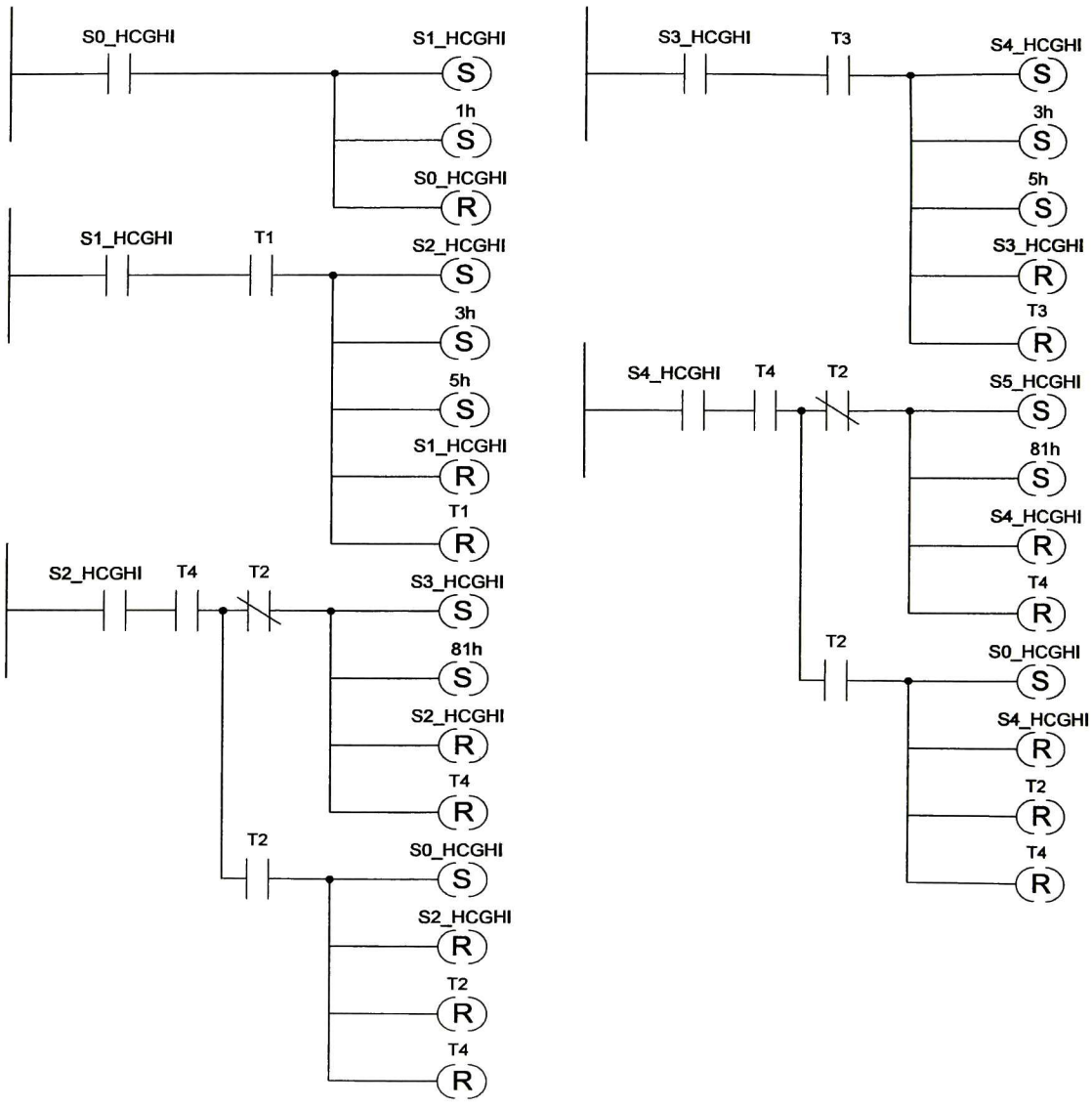


Figura B.4: Implementación en LD del Supervisor del nivel superior (HCGHI) parte 1

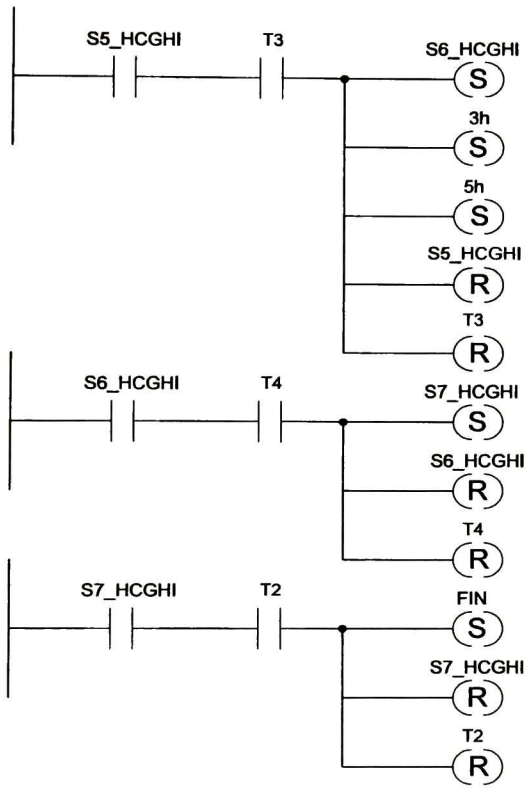


Figura B.5: Implementación en LD del Supervisor del nivel superior (HCGHI) parte 2

Bibliografía

[1] A. F. Vaz and W. M. Wonham. On supervisor reduction in discrete-event systems. *International Journal of Control*, 44(2):475–491, 1986.

[2] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.

[3] T. Ushio. A necessary and sufficient condition for the existence of finite state supervisors in discrete-event systems. *IEEE Transaction on Automatic Control*, 38(1):135–138, 1993.

[4] F. Lin and W. Wonham. On the computation of supremal controllable sublanguages. In *Proceedings 23rd Allerton Conference*, 1985.

[5] S. Lafortune and E. Chen. The infimal closed controllable superlanguage and its application in supervisory control. *IEEE Transactions on Automatic Control*, 35(4):398–405, April 1990.

[6] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete-event processes. *SIAM Juor. Control and Optimization*, 25(1):1202–1218, September 1987.

[7] F. Lin and W. M. Wonham. Decentralized control and coordination of discrete-event systems. In *Proceedings of the 27th Conference on Decision and Control Austin, Texas*, December 1998.

[8] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.

[9] F. Lin and W. M. Wonham. Decentralizaced control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 35(12):1330–1335, December 1990.

[10] K. Rudie and W. Wonham. Think globally, act locally: Decentralized discrete-event control problems. *IEEE Transaction on Automatic Control*, 40:1313–1318, 1995.

[11] W. M. Wonham. Supervisory control of discrete event systems. Available in *www.control.utoronto.ca/ ~ wonham*, 2009.

[12] H. Zhong and W. M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35:1125–1134, 1990.

[13] H. Zhong. *HierarchicaI control of discrete-event systems.* PhD thesis, Department of Electrical Engineering, University of Toronto, 1992.

[14] K.C. Wong and W.M. Wonham. *Hierarchical Control of Discrete–Event Systems.* Kluwer Academic, 1996.

[15] R J. Leduc, M. Lawford, and P. Dai. Hierarchical interface–based supervisory control of a flexible manufacturing system. *IEEE Transactions on Control Systems Technology*, 14(4):654–668, July 2006.

[16] R.C. Hill, J.E.R. Cury amd M.H. de Queiroz, D.M. Tilbury, and S. Lafortune. Multi–level hierarchical interface–based supervisory control. *Automatica*, 46:1152–1164, 2010.

[17] F. Jaimes. Modelado y control de sistemas automatizados de manufactura. Tesis de Maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2004.

[18] A. Sanchez, E. Aranda-Bricaire, F. Jaimes, E. Hernandez, and A. Nava. Synthesis of product–driven coordination controllers for a class of discrete–event manufacturing systems. *Robotics and Computer–Integrated Manufacturing*, 26:361–369, 2010.

[19] M. Uzam, G. Gelen, and R. Dalci. A new approach for the ladder logic implementation of ramadge-wonham supervisors. *Information, Communication and Automation Technologies*, 1:1–7, 2009.

[20] T. Krairojananan and S. Suthapradit. A plc program generator incorporating sequential circuit synthesis techniques. *Circuits and Systems IEEE APCCAS*, 1:399 – 402, 1998.

[21] K. W. Leucht and G. S. Semmel. Automated translation of safety critical application software specifications into plc ladder logic. In *Aerospace Conference IEEE*, 2008.

[22] L. Ferrarini, M. Romano, and C. Veber. Automatic generation of awl code from iec 61499 applications. *Industrial Informatics, IEEE International Conference on*, 1:25–30, August 2006.

[23] J. Richardson and M. Fabian. Automatic generation of plc programs for control of flexible manufacturing cells. *Emerging Technologies and Factory Automation. Proceedings. ETFA '03. IEEE Conference*, 1:337–344, 2003.

[24] J. Huang and R. Kumar. Nonblocking directed control of discrete event systems. *Decision and Control and European Control Conference. CDC-ECC '05. 44th IEEE Conference on 12-15 Dec.*, 1:7627 – 7632, 2005.

[25] P.J. Gawthrop, E.W., and McGookin. Using LEGO in control education. In *7th IFAC Symposium on Advances in Control Education. Madrid, Spain*, pages 1230–1235, 21–23 June 2006.

[26] D. Benedettelli, M. Casini, A. Garulli, A. Giannitrapani, and A. Vicino. A LEGO mindstorms experimental setup for multi-agent systems. In *Proc. of Intl. Symposium onControl Applications and Intelligent Control, (ISIC), Saint Petersburg*, pages 1230–1235, July 2009.

[27] E. Hernandez. Modelado jerárquico-modular y control de recursos de coordinación para sistemas de manufactura flexibles. Tesis de Maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2005.

[28] J. A. Nava. Marco de modelado para diseñar la capa de coordinación en arquitecturas de control industrial. Tesis de Maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2006.

[29] F. A. Garcia. Desarrollo formal de controladores lógicos. aplicación a un caso de estudio en sistemas de manufactura. Tesis de Maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2007.

[30] L. E. Llamas. Síntesis de arquitecturas de coordinación en sistemas automatizados de manufactura. Tesis de Maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2008.

[31] H. Rivera. Estudio comparativo de estrategias de control jerárquico-modular para sistemas automatizados de manufactura. Tesis de Maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2007.

[32] J.E. Tapia Medina. Desarrollo de una interfaz electrónica para el control de un prototipo de sistema automatizado de manufactura. Technical report, Universidad Tecnológica de la Zona Metropolitana de Guadalajara, 2007.

[33] P.J.G. Ramadage and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 2002.

[34] L. Yang and W.M. Wonham. Controllability and observability in the state-feedback control of discrete-event systems. In *Decision and Control*, volume 1, pages 203–208. 27th IEEE Conference, 1988.

[35] K. Akesson, M. Fabian, H. Flordal, and R. Malik. Supremica, An integrated environment for verification, synthesis and simulation of discrete event systems. In *Proc. 8th Workshop of Discrete Event Systems (WODES)*, Ann Arbor, Michigan 2006.

[36] J. Karl-Heinz and M. Tiegelkamp. Iec 61131-3: Programming industrial automation systems. concepts and programming languages, requirements for programming systems, decision-making aids. Springer, April 2001.

[37] Practical industrial programming using lec 61131-3 for plcs. Available on www.idc-online.com.

[38] ISA-88.01. Batch Control Systems. Part 1. Models and terminology. Standards. Technical report, Instrument Society of America, 1995.

[39] ISA-95.1. Enterprise - Control System Integration. Part 1: Models and Terminology. Standards. Technical report, Instrument Society of America, 1999.

[40] L. A. Bryan and E. A. Bryan. *Programmable Controllers. Theory and Implementation.* An Industrial Text Company Publication, 1997.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

"2011, Año del Turismo en México"

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Implementación Formal de Supervisores Jerárquico-Modulares en Aplicaciones de Manufactura

del (la) C.

Jorge Alfonso BUCIO CISNEROS

el día 09 de Agosto de 2011.

Dr. Arturo del Sagrado Corazón
Sánchez Carmona
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Andrés Méndez Vázquez
Investigador CINVESTAV 2A
CINVESTAV

Dr. Eduardo Aranda Bricaire
Investigador CINVESTAV
CINVESTAV