UNIVERSIDAD PRIVADA ANTENOR ORREGO

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

"PROPUESTA DE METODOLOGIA PARA PROGRAMACION DE PLC EN LENGUAJE LADDER"

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

AUTORES:Br. TULIO CÉSAR MORENO MORALES

ASESOR: ING. LINARES VÉRTIZ SAÚL N.

TRUJILLO - PERÚ

2018

ACREDITACIONES
TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

TÍTULO:

"PROPUESTA DE METODOLOGIA PARA PROGRAMACION DE PLC EN LENGUAJE LADDER"

AUTOR:

Br. TULIO CÉSAR MORENO MORALES
Tesista

APROBADO POR:

Ing. Filiberto Azabache Fernández
PRESIDENTE
N° CIP: 97916

Ing. Luis Alberto Vargas Díaz
SECRETARIO
N° CIP: 104175

Ing. Lenin Humberto Llanos León.
VOCAL
N° CIP: 139213

Ing. Saúl Noé Linares Vértiz
ASESOR
N° CIP:142213

PRESENTACIÓN

Señores Miembros del Jurado:

Dando cumplimiento y conforme a las normas establecidas en el Reglamento de Grados y Títulos y Reglamento de la Facultad de Ingeniería de la Universidad Privada Antenor Orrego, para obtener el título profesional de Ingeniero Electrónico, se pone a vuestra consideración el Informe del Trabajo de Investigación Titulado "PROPUESTA DE METODOLOGIA PARA PROGRAMACIÓN DE PLC EN LENGUAJE LADDER" con la convicción de alcanzar una justa evaluación y dictamen, excusándonos de antemano de los posibles errores involuntarios cometidos en el desarrollo del mismo.

Trujillo, 7 de Noviembre de 2018

DEDICATORIA

Para mis hijas Fátima y Carolina, por supuesto.

AGRADECIMIENTO

A Nelly Morales, por su constante empuje y sabios consejos.

RESUMEN

El presente trabajo de investigación está enfocado a facilitar la programación de los controladores lógicos programables “PLC”, usando la herramienta matemática del Algebra de Boole.

Esta investigación consta de 7 capítulos en los cuales se ha desarrollado el tema, en el primer capítulo se aborda la realidad problemática que se tiene con los ingenieros Junior al momento de programar los PLC. El capítulo 2 aborda los antecedentes de la investigación y el marco teórico necesario para la investigación. El capítulo 3 se indica la hipótesis del problema y se indican las variables de la investigación. El capítulo 4 muestra los objetivos de la investigación.

El capítulo 5 está destinado al desarrollo de la investigación, allí se prueban las herramientas matemáticas para la solución de programas en lenguaje Ladder.

El capítulo 6 indica los resultados de la investigación donde se indica la metodología para el desarrollo de programas en lenguaje Ladder.

El capítulo 7 está destinado a conclusiones y recomendaciones sobre la investigación.

ABSTRACT

The present work of investigation is focused to facilitate the programming of the programmable logical controllers "PLC", using the mathematical tool of the Boolean Algebra.

This investigation consists of 7 chapters in which the subject has been developed, in the first chapter the problematic reality that is had with Junior engineers when programming PLCs is addressed. Chapter 2 deals with the background of the research and the theoretical framework necessary for the investigation. Chapter 3 indicates the hypothesis of the problem and indicates the variables of the investigation. Chapter 4 shows the objectives of the investigation.

Chapter 5 is intended for the development of research, there are tested the mathematical tools for the solution of programs in Ladder language.

Chapter 6 indicates the results of the research where the methodology for the development of programs in Ladder language is indicated.

Chapter 7 is intended for conclusions and recommendations on research.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

1. EL PROBLEMA

1.1. Planteamiento del problema

Los Controladores Lógicos Programables son equipos electrónicos que permiten gran flexibilidad al momento de automatizar un proceso industrial, ello ha permitido reducir el espacio de los tableros de mando y además la posibilidad de poder cambiar la filosofía de control en cualquier momento con poco esfuerzo. Sin embargo, el entorno de programación de estos, está dada por cada fabricante ya sea Siemens, Rockwell, Schneider etc. algunos más amigables con respecto a otros pero que cumplen bien su función.

Los lenguajes que permiten la programación de estos PLC, son bastante conocidos, como por ejemplo Ladder, AWL, Bloques de funciones como se ilustra en la figura 1. Dentro de ellos el más conocido es el lenguaje Ladder. Ya que la mayor parte de programadores están familiarizados con la lógica de Relés y esto les permite adaptarse con mayor facilidad al entorno de programación.

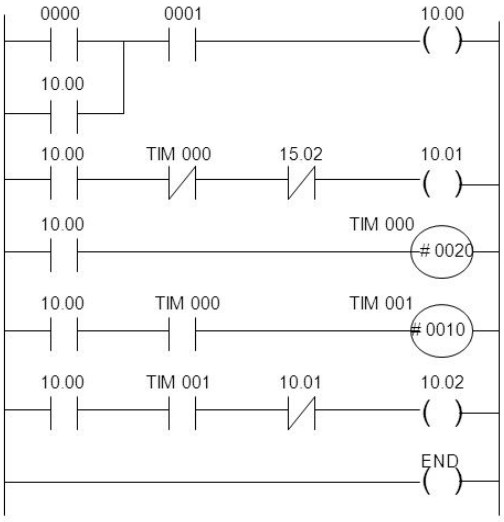

Figura 1: Lenguaje Ladder y AWL

Los desarrolladores de las soluciones de automatización, por lo general al momento de realizar una automatización de un proceso por lo general el primer paso es recopilar el requerimiento del proceso a automatizar, luego determinan las variables de entrada y finalmente las variables de salida que controlaran, luego se procede al diseño del algoritmo que permitirá el control de ese proceso, para ello el hace uso de su experiencia en la programación de procesos parecidos que le dieron buen resultado y junta estos sub procesos con el afán de dar solución al requerimiento de la automatización, pero ello lleva a que en algunos casos los algoritmos no sean los más adecuados llegando a ser muy extensos y ocupando grandes porciones de memoria en los PLC, en otros casos lograr la realización de los algoritmos lleva mucho tiempo ya que en nuestra experiencia nunca nos encontramos con un caso parecido y por lo tanto es algo nuevo que se tiene que explorar, en el peor de los casos no se puede realizar por la carencia de experiencia. Lo expuesto anteriormente se verifica en las empresas de nuestro medio al momento de realizar una automatización, cuando no pueden realizar

alguna función recurren a la experiencia de los expertos desarrolladores de las respectivas marcas, aumentando de esta manera la dependencia de la experiencia.

1.2. Definición del problema

Realización de algoritmos de control para PLC basados en experiencias anteriores.

Formulación del problema:

¿Cómo independizar la experiencia en la realización de algoritmos de control para PLC?

Alcance:

Esta investigación abarca la propuesta de una metodología que permita la programación de PLC sin tener en cuenta la experiencia en programación de procesos industriales.

1.3. Justificación de la investigación

En lo académico:

Permitirá a los Docentes proponer una solución a la falta de experiencia en la programación de PLCs.

1.4. Aportes

En lo académico, brindar una nueva propuesta para la programación de los PLCs que permita a los nuevos desarrolladores dar soluciones confiables sin necesidad de tener experiencia en programación de procesos industriales.

2. MARCO TEORICO

2.1. Antecedentes de la Investigación

Aparicio, C. (2008) "***Estructuración de sistemas de control de eventos discretos en un plc aplicado a procesos híbridos***", Mexico: Instituto Tecnológico y de Estudios Superiores de Monterrey

En este trabajo se presenta un resumen del marco teórico de los sistemas híbridos; su definición formal, así como aplicaciones. Así mismo, se presenta una simulación de un caso práctico, el proceso de producción del concreto, investigación bibliográfica del proceso, y la formulación de un modelo híbrido para analizar y controlar. Así, mediante la aplicación de un sistema de control discreto sobre las variables continuas y discretas del proceso, se busca mejorar el desempeño del mismo.

Boscán, L. (2010) **"Diseño de un sistema de control mediante PLC para las instalaciones de aire acondicionado central (agua helada) e iluminación de un edificio de laboratorios".**Caracas: Universidad Central de Venezuela.

El presente trabajo se presenta el diseño de un sistema para la automatización de los sistemas de aire acondicionado y luminarias del edificio de laboratorios Lab –Volt ubicado en el Vigía- Edo. Mérida, con la finalidad de mejorar las condiciones de confort deledificio para los usuarios controlando la temperatura de los espacios manteniendo una temperatura de 74ºF y reducir el consumo de energía eléctrica de éste mediante la programación horaria y la colocación de detectores de presencia, que controlan el encendido iluminación y aire acondicionado. Mediante la correcta selección de los actuadores y sensores a utilizar para el control, la elección del Controlador Lógico Programable Telemecanique modelo Twido,

Trejo, S., Tejada D. (2014). **"Diseño de un sistema automático e instrumentación para la planta de almacenamiento y despacho de petróleo de la empresa Olympic Perú-Piura**". Trujillo, Peru. Universidad Privada Antenor Orrego,

El presente trabajo de investigación fue desarrollado teniendo en cuenta el problema planteado por la empresa Olympic Perú, para lo cual fue necesario utilizar los conocimientos adquiridos en control y automatización, además de estándares para manejo de hidrocarburos, se realizó el análisis matemático, obtención de la función de transferencia, se propone el plano de instrumentación y también se propone modelo y marca para cada instrumento detallado en el plano de instrumentación

Vega, A. (206).***"Diseño de un Circuito Logico para Funtores Logicos"***. Mexico. Universidad Naciona Autonoma de Mexico.

En el presente trabajo describe la implementación de Funtores lógicos usando el Algebra de Boleana , se hacen algunas consideraciones que se deben tomar al momento de diseñar en la plataforma de desarrollo

2.2. Controlador Lógico Programable (PLC)

2.2.1. Definición

Según la Asociación Nacional de Fabricantes Eléctricos de Estados Unidos "Es un dispositivo digital electrónico con una memoria programable para el almacenamiento de instrucciones, permitiendo la implementación de funciones específicas como ser: lógicas, secuenciales, temporizadas, de conteo y aritméticas; con el objeto de controlar máquinas y procesos."

2.2.2. Estructura de un PLC

La estructura básica de un PLC está compuesta por:

- ✓ *Fuente de alimentación*

  La fuente de alimentación proporciona las tensiones necesarias para el funcionamiento de los distintos circuitos del sistema. La alimentación a la CPU frecuentemente es de 24 Vcc, o de 110/220 Vca.

- ✓ *CPU.*

  Es la parte inteligente del sistema. Interpreta las instrucciones del programa de usuario y consulta el estado de las entradas. Dependiendo de dichos estados y del programa, ordena la activación de las salidas deseadas.

- ✓ *Periféricos de entradas y salidas.*

  Entradas: Corresponde al elemento o interfaz por el cual ingresan los datos que son adaptados y codificados en forma comprensible para la CPU.

  Salidas: Trabaja con las señales entregadas de la CPU, decodificándolas y amplificándolas para manejar distintos tipos de actuadores

- ✓ *Memorias*

  Esta etapa es la encargada de almacenar la información del programa y los datos con los cuales trabaja la CPU. Dependiendo de la función se utilizarán distintos tipos de memoria, como por ejemplo: memoria de usuario, memoria de tabla de datos, memoria de sistema y memoria de almacenamiento.

  Memoria ROM: Es una memoria de sólo lectura que contiene el sistema operativo (firmware) con el que opera el controlador.

  Memoria RAM: Es una memoria volátil y de aplicación, ya que en ésta se ubica el programa de usuario.

- ✓ *Unidad de programación*

  Es un terminal a modo de ordenador que proporciona una forma más favorable de realizar el programa de usuario y observar parámetros internos del autómata.

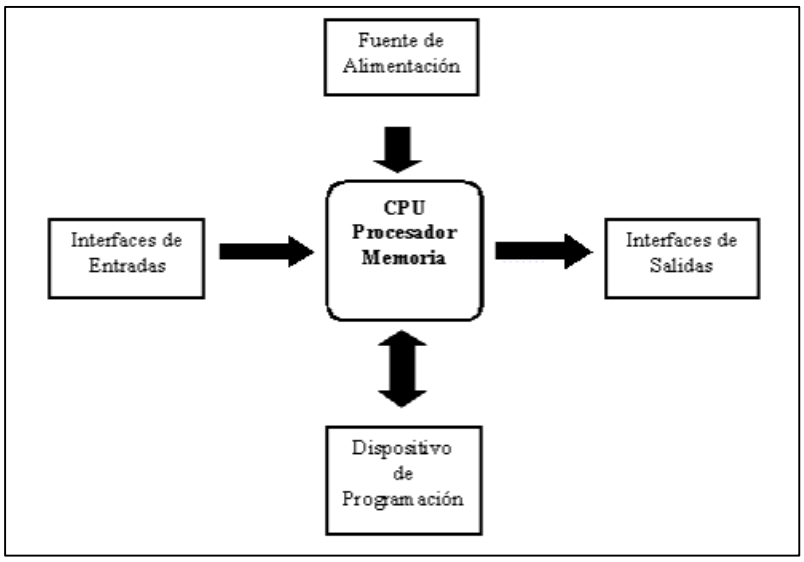

Figura 2: Estructura básica de un PLC

2.2.3. Ciclo de ejecución de un PLC

Al iniciar el ciclo la CPU lee el estado de las entradas, posteriormente ejecuta la aplicación empleando el último estado leído. Una vez completado el programa, la CPU ejecutará tareas internas de diagnóstico y comunicación. Al final del ciclo se actualizan las salidas. El tiempo de ciclo depende del tamaño del programa, del número de E/S y de la comunicación requerida. En la figura 3, se muestra el ciclo de ejecución de un Controlador lógico programable.

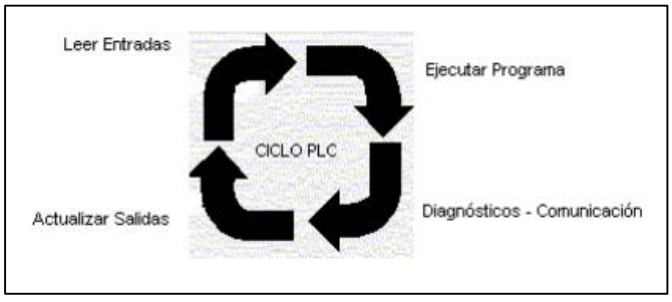

Figura 3: Ciclo de ejecución por el PLC

2.2.4. Lenguajes de programación para PLC

- ✓ *KOP*

Sigue los principios del lenguaje “Esquema de contactos” (en inglés Ladder Logic). En el lenguaje KOP, las operaciones lógicas con bits operan con dos dígitos, 1 y 0. Estos dos dígitos constituyen la base de un sistema numérico denominado sistema binario. En el ámbito de los contactos y bobinas, un 1 significa activado ("conductor") y un 0 significa desactivado ("no conductor"). La nomenclatura que utiliza este lenguaje es el siguiente:

- ---| |--- Contacto normalmente abierto
- ---| / |--- Contacto normalmente cerrado
- ---( ) Bobina de relé, salida

Las operaciones lógicas con bits interpretan los estados de señal 1 y 0, y los combinan de acuerdo con la lógica de Boole. Estas combinaciones producen un 1 o un 0 como resultado y se denominan "resultado lógico" (RLO). Las operaciones lógicas con bits permiten ejecutar las más diversas funciones.

✓ *AWL*

El lenguaje de programación AWL (lista de instrucciones) es un lenguaje textual orientado a la máquina. Las diversas instrucciones equivalen a los pasos de trabajo con los que la CPU ejecuta el programa y éstas se pueden reunir en segmentos. Con este lenguaje editar bloques S7 de forma incremental o crear su programa en una fuente AWL con un editor orientado a la fuente para compilarlo luego en bloques.

Las instrucciones AWL se dividen en:

- OPERACION: indica la instrucción que se ha de realizar (ej. AND).
- OPERANDO: indica una constante o dirección con la que debe trabajar la operación. Si se trata de una dirección se puede manejar en modo bit, byte o palabra.

✓ *FUP*

Sigue los principios del lenguaje “Diagrama de funciones” fijados en la norma DIN EN-61131-3 (int. IEC 1131-3). Es un lenguaje gráfico que utiliza los cuadros del álgebra booleana para representar la lógica. Asimismo, permite representar funciones complejas (por ejemplo, funciones matemáticas) mediante cuadros lógicos. Cuando hay mucha lógica booleana en serie suele ser más compacto y más fácil de ver el segmento completo

2.3. Algebra Booleana.

2.3.1. Introducción.

En 1815 George Boole propuso una herramienta matemática llamadá Algebra de Boole.

Luego en 1938 Claude Shannon propuso que con esta´ algebra es posible modelar los llamados Sistemas Digitales

Los circuitos eléctricos simples son circuitos conmutados con interruptores que están conformados por una conexión de una fuente de voltaje (V), un interruptor o suiche (S) y una bombilla o lámpara (LAMP). La función de este sistema eléctrico consiste en cerrar o abrir el interruptor para que se encienda o se apague la lámpara, respectivamente.

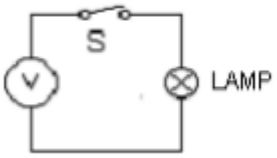

Figura 4: Circuito eléctrico simple

Al cerrar o poner el interruptor en estado lógico1 “1”, se produce el encendido de la lámpara, a esta acción se asignará “1”; al abrir o poner el interruptor en estado en lógico “0”, se produce el apagado de la lámpara, a cuya acción se asignará el estado lógico “0”

S	LAMP
0	0
1	1

Tabla N° 1: Circuito Eléctrico simple

2.3.2. Tipos de circuitos Conmutados.

Los circuitos conmutados según la distribución de sus interruptores se pueden clasificar así:

2.3.2.1. Circuitos conmutados en serie.

Son aquellos circuitos cuyos interruptores van de manera consecutiva

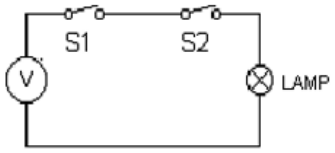

Figura 5: Circuitos conmutados en serie

2.3.2.2. Circuitos conmutados en paralelo.

Son aquellos circuitos cuyos interruptores van distribuidos en diferentes filas

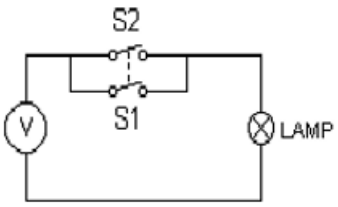

Figura 6: Circuito conmutado en paralelo

2.3.3. Funciones lógicas de circuitos conmutados.

Una tabla de verdad contiene valores obtenidos de las posibles combinaciones de los valores de los interruptores o pulsadores (o simplemente entradas) también se denomina función lógica.

Según la distribución de los interruptores o pulsadores en el circuito en serie corresponde a la función lógica denominada función AND y equivale al encendido o apagado de la lámpara. Ahora, la distribución de los interruptores o pulsadores en el circuito en paralelo se produce la o función OR y corresponde al encendido o apagado de la lámpara.

2.3.3.1. Función lógica de un circuito conmutado en serie.

Un circuito conmutado en serie tiene como función, dar encendido a la lámpara, si los interruptores están cerrados o apagarla, si se abre algún interruptor o pulsador. La función lógica de este circuito corresponde a la compuerta AND

Función AND		
S1	S2	LAMP
0	0	0
0	1	0
1	0	0
1	1	1

Tabla N° 2: Función lógica de un circuito conmutado en serie

2.3.3.2. Función lógica de un circuito conmutado en paralelo.

Un circuito conmutado en paralelo tiene como función darle encendido a la lámpara, siempre que alguno de los interruptores esté cerrado o de apagarla cuando se abren todos los interruptores o pulsadores. La función lógica de este circuito corresponde a la compuerta OR

Función OR		
S1	S2	LAMP
0	0	0
0	1	1
1	0	1
1	1	1

Tabla N° 3: Función lógica de un circuito conmutado en paralelo

3. HIPOTESIS

3.1. General

El desarrollo de una metodología basada en algebra de Boole permitirá la realización de algoritmos para automatización de procesos.

Variable independiente:

Metodología.

Variable dependiente:

Algoritmo de Automatización.

3.2. Operacionalización de las variables

Tabla N° 04: Operacionalización de la variable independiente

VARIABLE	DEFINICION CONCEPTUAL	DEFINICION OPERACIONAL	INDICADORES	INSTRUMENTO	FORMULA	UNIDADES DE MEDIDA
Metodología	Grupo de mecanismos o procedimientos racionales, empleados para el logro de un objetivo	Procedimiento para la realización de un Algoritmo	Numero etapas.	Reporte de simulación.	----------	
			Herramienta matemática.	Reporte de Diseño	----------	

Tabla N° 4: Operacionalización de la variable independiente. Fuente: Elaboración propia

Tabla *N°05:* Operacionalización de la variable dependiente

VARIABLE	DEFINICION CONCEPTUAL	DEFINICION OPERACIONAL	INDICADORES	INSTRUMENTO	FORMULA	UNIDADES DE MEDIDA
Algoritmo de Automatización.	Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas	Conjunto de operaciones para realizar una automatización.	Numero de variables intervinientes	Reporte Simulación	----------	
			Numero de bloques de programación	Reporte simulación	----------	

Tabla N° 5: Operacionalización de la variable dependiente. Fuente: Elaboración propia

4. OBJETIVOS

4.1. General

- Diseñar una metodología para la elaboración de un algoritmo de programación de PLC en lenguaje Ladder.

4.2. Específicos

- Determinar la herramienta matemática para modelar los sistemas discretos.
- Determinar las técnicas que permitan la simplificación de funciones discretas.
- Elaborar el algoritmo de programación

5. MATERIAL Y PROCEDIMIENTOS

5.1. Población y muestra.

Población:
Programas de automatización con PLC
Muestra:
Programas de automatización con PLC discretos.

5.2. Metodología.

5.2.1. CIRCUITO SERIE

Cuando se tiene dos o más variables discretas en forma de producto, su equivalente circuital es una conexión en serie.

$$F(X,Y,Z) = XYZ$$

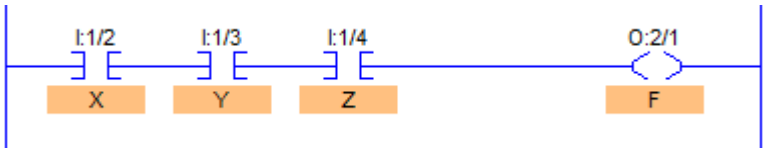

Figura 7: Circuito Serie; Elaboración: fuente propia

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabla N° 6: Función para circuito Serie tres variables

5.2.2. CIRCUITO PARALELO

Si las variables de una función booleana se suman circuitalmente indica que estas se encuentran en paralelo.

$$F(X,Y) = X + Y$$

Figura 8: Circuito Paralelo; Elaboración: fuente propia

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

Tabla N° 7: Función para circuito en paralelo.

5.2.3. COMPLEMENTO

Cuando una variable se complementa indica que el interruptor esta normalmente cerrado.

$$F(X) = X'$$

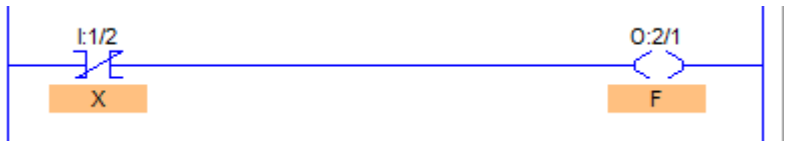

Figura 9: Interruptor normalmente cerrado; Elaboración: fuente propia

X	F
0	1
1	0

Tabla N° 8: Función complemento.

5.2.4. IMPLEMENTACION

La implementación de las funciones booleanas se realiza haciendo uso de las combinaciones anteriores, por ejemplo sea $f(x, y) = (x + y)x'$
Circuitalmente será:

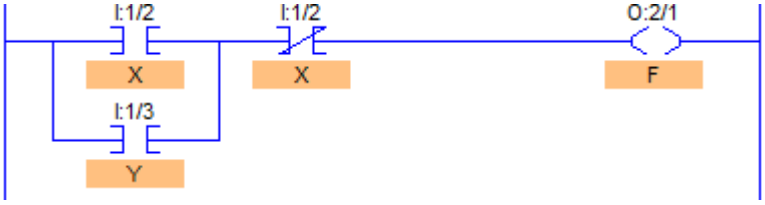

Figura 10: Circuito implementación; Elaboración: fuente propia

Y su tabla de verdad será.

x	y	$f(x,y)=(x+y)x'$
0	0	0
0	1	1
1	0	0
1	1	0

Tabla N° 9: Función f(x,y)= (x+y)x'

5.2.5. SIMPLIFICADO

La simplificación se puede hacer de 2 maneras, una de ellas es a través del uso de Algebra Booleana**,** lo que involucra conocer muy bien las propiedades del Algebra de Boole, esto puede ser algo desventajoso al momento de llevarlo a la práctica pues los problemas son en algunos casos extremadamente largos y por ende tediosos.
Simplifiquemos la siguiente función $f(x,y)=(x+y)x'$ usando Algebra de Boole.

$$f(x,y)=xx'+x'y=0+x'y=x'y$$
$$f(x,y)=x'y$$

Si ahora simplificamos usando mapas de karnaugh se debe tener en cuenta los términos mínimos o máximos, de acuerdo al punto 5.2.1 se tiene que.

$$f(x,y)=\sum(1)=\prod(0,2,3)$$

Por lo tanto, el mapa será.

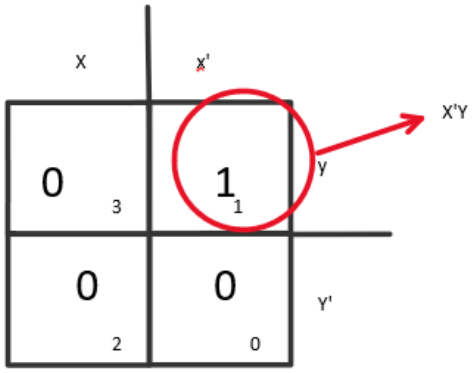

Figure 11 : Simplificación usando mapas de Karnaugh f(x,y)=(x+y)x'

$$f(x,y) = x'y$$

Se puede ver la potencia de la herramienta en el siguiente ejemplo.
Simplificar el siguiente esquema.

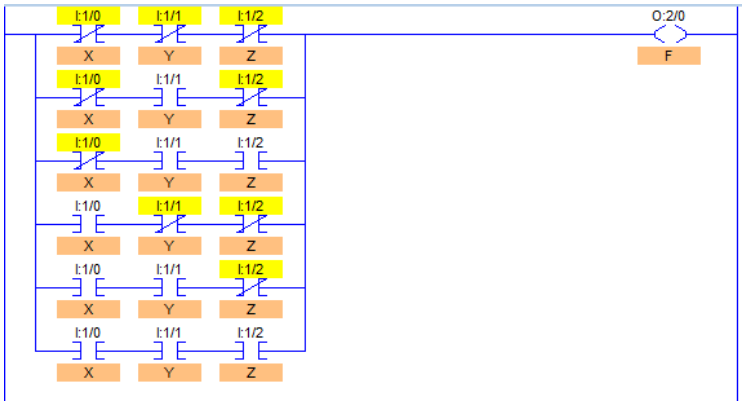

Figure 12: Implementación de esquema a simplificar

$$f(x,y,z) = x'y'z' + x'yz' + x'yz + xy'z' + xyz' + xyz$$

$$f(x,y,z) = \sum(0,2,3,4,6,7)$$

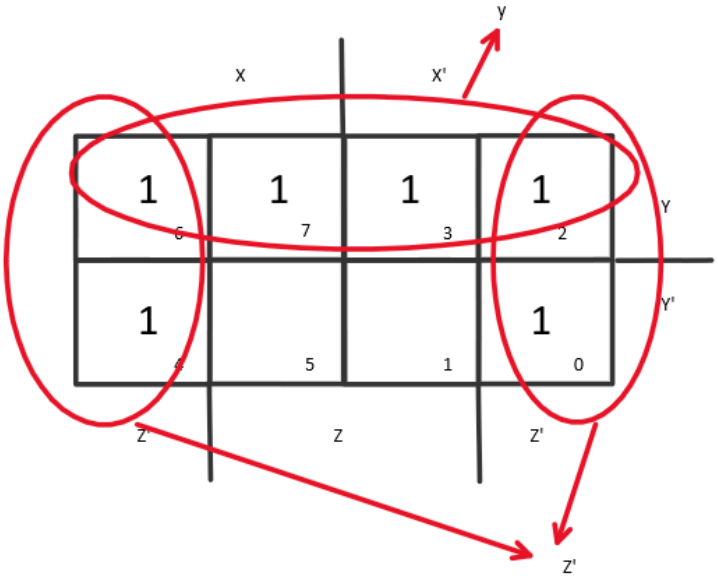

Figure 13: Simplificación usando Mapas de Karnaugh

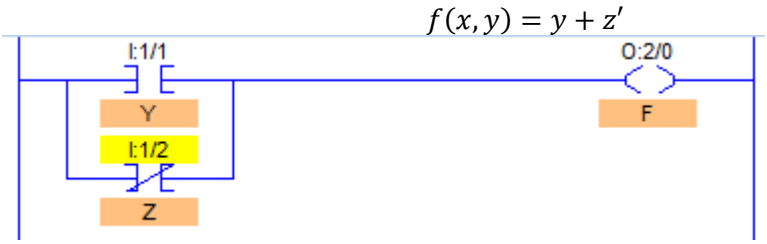

Figure 14: Implementación de esquema ya simplificado

5.2.6. Modelamiento de procesos a partir de condiciones deseadas.

Se desea activar un motor con 2 interruptores, cada uno de ellos tiene la capacidad de activar o desactivar al motor independientemente del estado en que se encuentre el interruptor.
Para desarrollar este esquemático daremos nombres a cada interruptor.
Sea el interruptor 1 "x" y el interruptor 2 "y", el estado de reposo será "0" , que indica que el motor está apagado
Partiremos de una condición inicial.

Termino	x	y	Motor
0	0	0	0

Tabla N° 10: Condición inicial activar/desactivar motor con dos interruptores

Si a partir de esta condición algún interruptor es pulsado el motor se activará.

Termino	x	y	Motor
1	0	1	1
2	1	0	1

Tabla N° 11: Condición activar motor con dos interruptores

Cuando el motor está activado si alguno de los interruptores cambia de estado el motor se apagara, esto sugiere que.

Termino	x	y	Motor
0	0	0	0
3	1	1	0

Tabla N° 12: Condición desactivar motor con dos interruptores

Así la función booleana será.

$$f(x,y) = \sum(1,2) = \prod(0,3)$$

Termino	x	y	Motor
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Tabla N° 13: Función del motor: activar/desactivar motor con dos interruptores

Usando el mapa para la simplificación se observa que este no se puede simplificar.

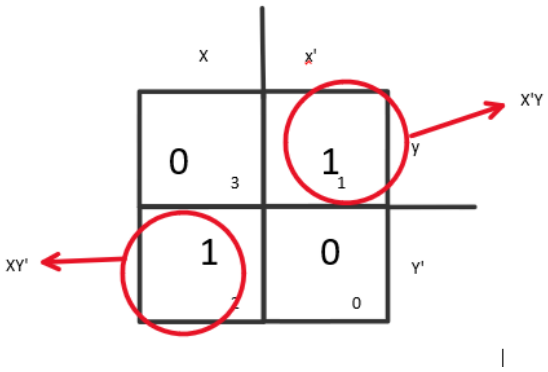

Figure 15: Simplificación activación/apagado de motor con 2 interruptores usando mapa de Karnaugh

$$f(x,y) = x'y + xy'$$

La implementación será la siguiente.

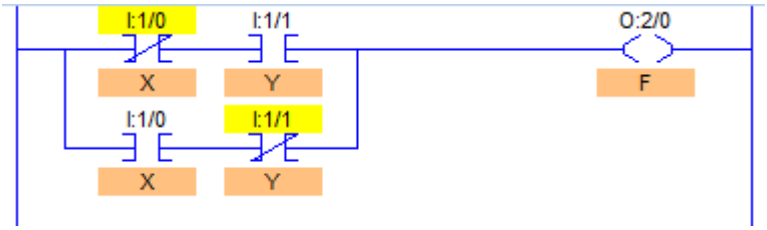

Figure 16: Implementación activación/apagado de motor con 2 interruptores

5.2.7. Uso de variables de estado.

Desarrollar un programa que permita el enclavamiento de una salida usando 2 pulsadores, el primero activa la salida y el segundo al ser pulsado la desactiva.
Para desarrollar este esquema se necesitara de tres variables

- Pulsador de Start
- Pulsador de Stop
- Salida.

Iniciaremos la solución indicando una condición inicial.

a) Los pulsadores no están presionados y la salida está apagada.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
0	0	0	0	0

Tabla N° 14: Condición inicial enclavamiento de una salida usando dos interruptores

b) Basado en las condiciones anteriores si se pulsa Start la salida se activará.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
2	0	1	0	1

Tabla N° 15: Presiona Start enclavamiento de una salida usando dos interruptores

c) Luego se suelta el pulsador de Start y la salida debe de seguir activada.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
4	1	0	0	1

Tabla N° 16: Suelta Start enclavamiento de una salida usando dos interruptores

d) Si ahora se presiona Stop como la salida esta activada esta debe de pasar al estado de apagado.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
5	1	0	1	0

Tabla N° 17: Presiona Stop enclavamiento de una salida usando dos interruptores

e) Si se suelta el pulsador de Stop la salida debe de matener su estado de apagado.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
0	0	0	0	0

Tabla N° 18: Suelta Stop enclavamiento de una salida usando dos interruptores

f) Si el sistema está apagado y se presiona Stop este debe permanecer apagado.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
1	0	0	1	0

Tabla N° 19: Sistema apagado presiona Stop enclavamiento de una salida usando dos interruptores

g) Si la salida esta activada y se presiona Start la salida debe de permanecer activada.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
6	1	1	0	1

Tabla N° 20: Sistema activado presiona Start enclavamiento de una salida usando dos interruptores

h) Si se presiona Start y Stop a la vez, independientemente del estado de la salida esta debe de pasar al estado de apagado.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
7	1	1	1	0
3	0	1	1	0

Tabla N° 21: Presionar Start y Stop´ enclavamiento de una salida usando dos interruptores

Con los datos obtenidos formaremos la función.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
0	0	0	0	0
2	0	1	0	1
4	1	0	0	1
5	1	0	1	0

1	0	0	1	0
6	1	1	0	1
7	1	1	1	0
3	0	1	1	0

Tabla N° 22: Función de la salida enclavamiento de una salida usando dos interruptores

$$f(salida, Satrt, Stop)] = \sum(2,4,6) = \prod(0,1,3,5,7)$$

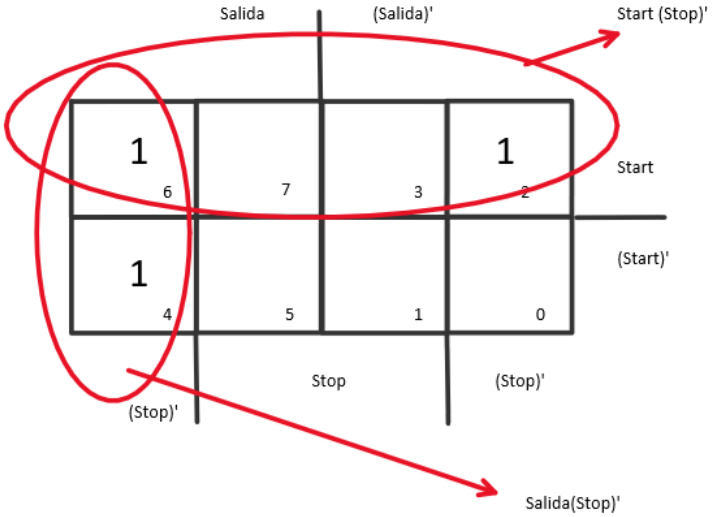

Figure 17: Simplificación de Enclavamiento de salida usando dos pulsadores usando mapa de Karnaugh

$$Salida = Start\ (Stop)' + Salida\ (Stop)'$$
$$Salida = (Start + Salida)(Stop)'$$

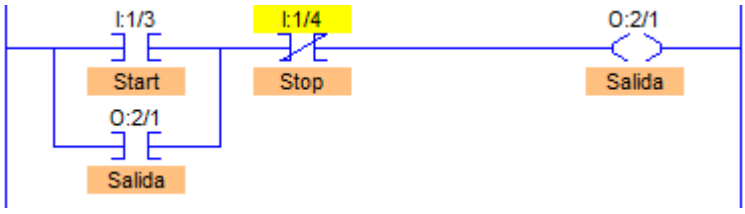

Figure 18: Implementación Enclavamiento de salida usando dos pulsadores

Pero esto se puede representar de otra manera.

$$Salida = (Stop)'(Start + Salida)$$

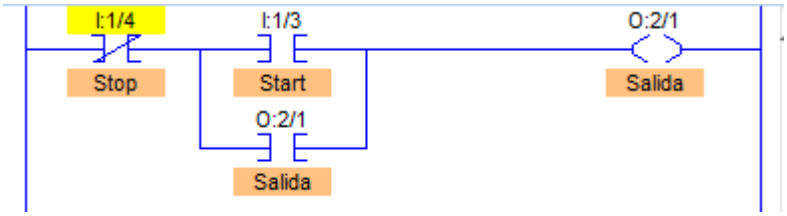

Figure 19: Implementación Enclavamiento de salida usando dos pulsadores segunda opción

Ambos resultados son correctos.
Si el resultado del punto "h" fuera verdadero, es decir al presionar Start y Stop a la vez, estos harán que se activara el motor. La función cambiara así.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
7	1	1	1	1
3	0	1	1	1

Tabla N° 23: Presiona Start y Stop para condición de salida activada enclavamiento de una salida usando dos interruptores

$$f(salida, Satrt, Stop)] = \sum(2,3,4,6,7) = \prod(0,1,5)$$

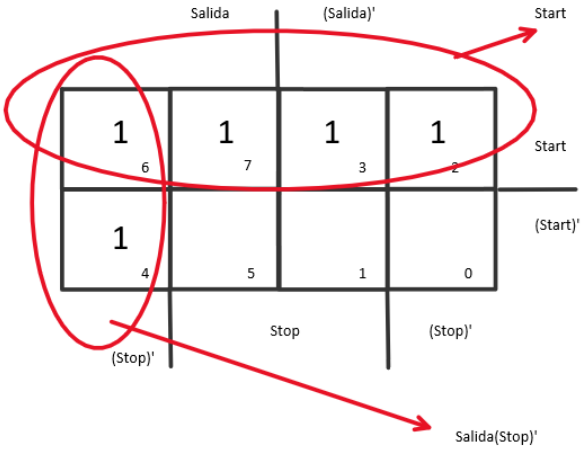

Figure 20: Simplificación de Enclavamiento de salida usando dos pulsadores usando mapa de Karnaugh

$$Salida = Start + Salida\ (Stop)'$$

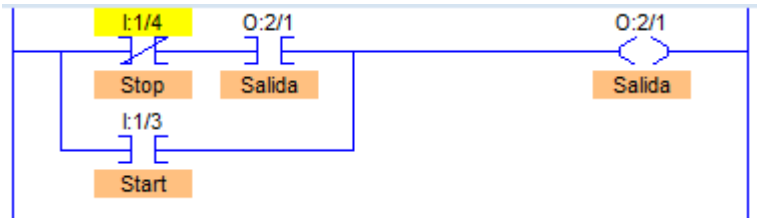

Figure 21; Implementación Enclavamiento de salida usando dos pulsadores

Esto indica una forma alternativa del enclavamiento conocido.

5.2.8. Manejo de Diagramas de Tiempo.

Los diagramas de tiempo están asociados al uso de temporizadores tanto con retardo a la conexión como con retardo a la desconexión.
Para ilustrar la herramienta Booleana desarrollaremos el siguiente ejercicio.
Diseñar un diagrama de mando para arrancar 2 motores en secuencia usando un botón de Start y parar ambos con un botón de Stop.
El primer motor debe de arrancar cuando se presione Start
El segundo motor debe de arrancar 5 segundos después de arrancar el primero.

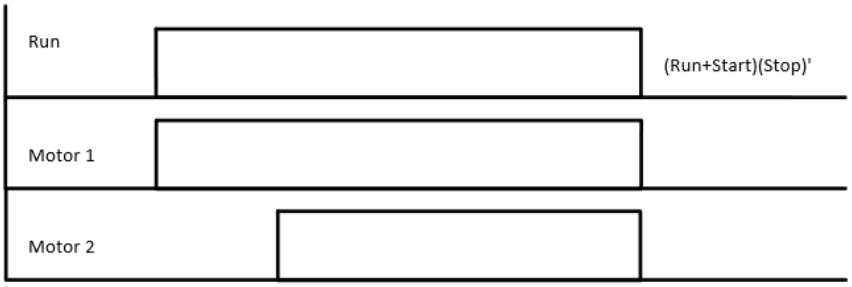

Figure 22: Diagrama de tiempo de 2 motores en secuencia usando un botón de Start y parar ambos con un botón de Stop

Para arrancar el motor 2 después de 5 segundos se necesitará un temporizador

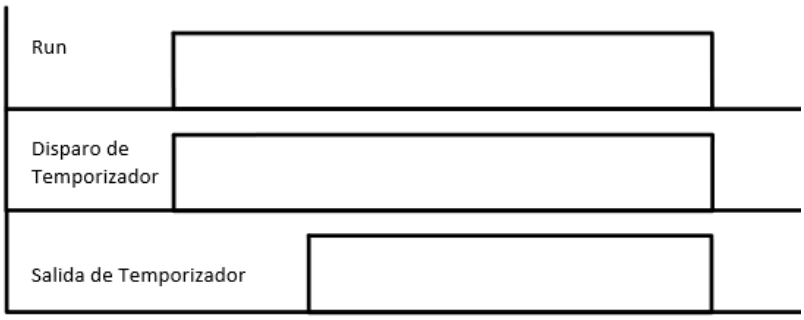

Figure 23: Diagrama de tiempo de disparo y salida de temporizador del segundo motor.

De la gráfica se observa que el disparo del temporizador es la variable "Run"

$$T_{in} = Run \quad donde \quad T_{in} = Disparo\ de\ Temporizador$$

Para calcular las funciones que gobiernan a los motores se tiene la siguiente gráfica.

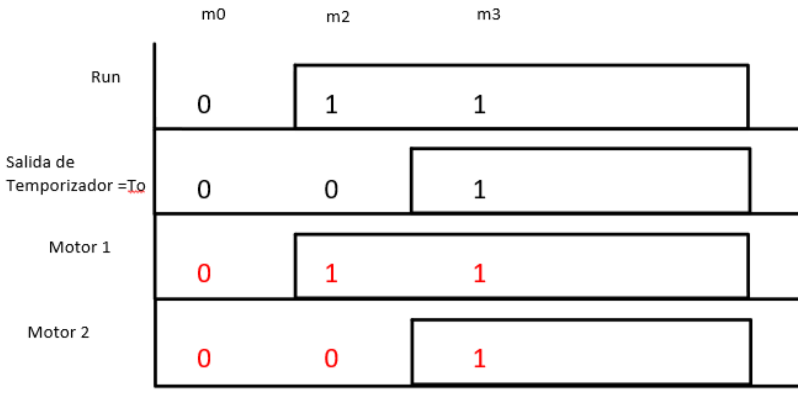

Figure 24: Grafica para calcular funciones que gobiernan a los motores

Se puede determinar las funciones para cada motor

$$Motor\ 1 = \sum(2,3) = \prod(0)$$

Y una condición aleatoria que es el termino mínimo "1"

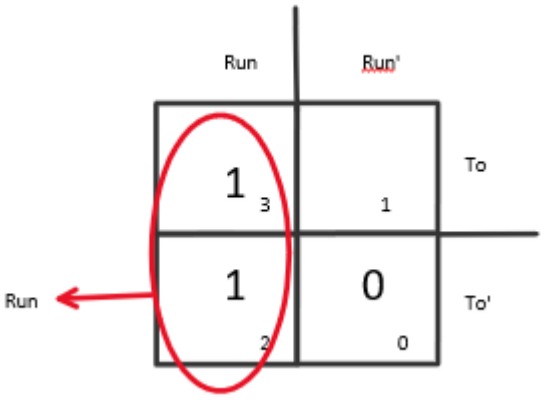

Figure 25: Simplificación función Motor 1

$$Motor\ 1 = Run$$

$$Motor\ 2 = \sum(3) = \prod(0,2)$$

Y una condición aleatoria que es el termino mínimo “1”

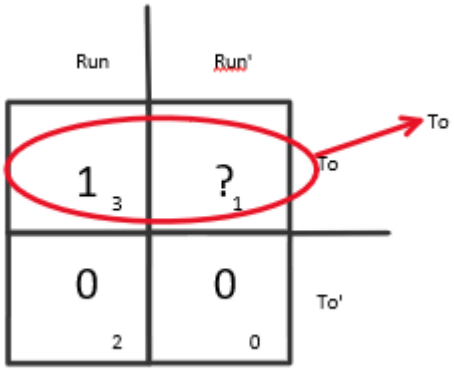

Figure 26: Simplificación motor 2

$$Motor\ 2 = To$$

Este resultado se pudo observar de la gráfica anterior de tiempos.

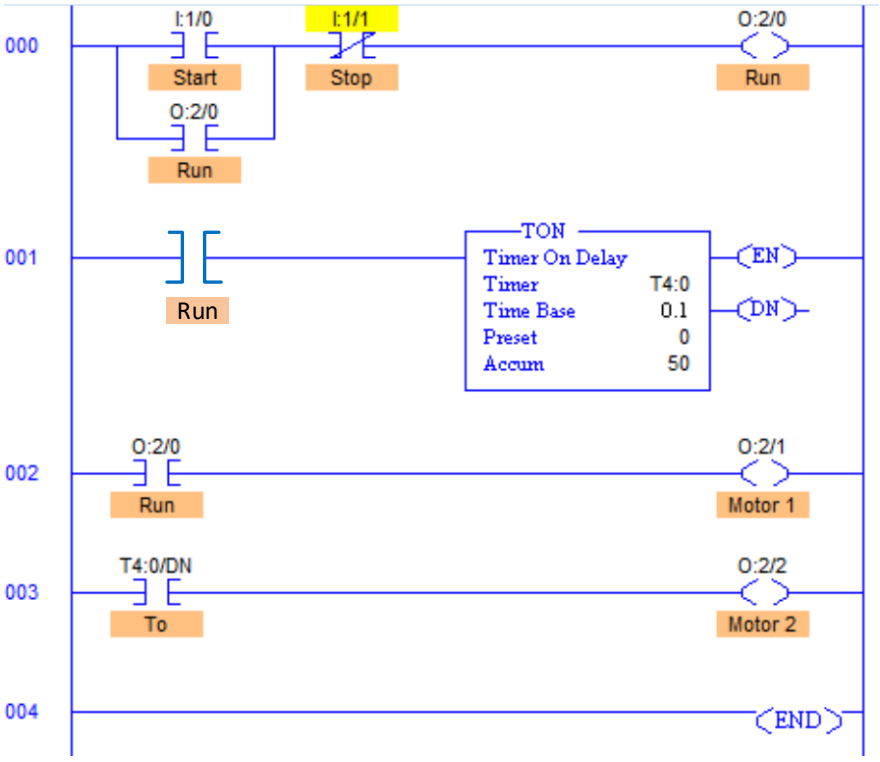

Figure 27: Implementación de mando para arrancar 2 motores en secuencia usando un botón de Start y parar ambos con un botón de Stop

5.2.9. Uso de variables auxiliares.

Desarrollar un diagrama de mando para el siguiente sistema.
Un motor se activa 5 segundos después de presionar Start, y se apaga luego de 6 segundos de presionado Stop.

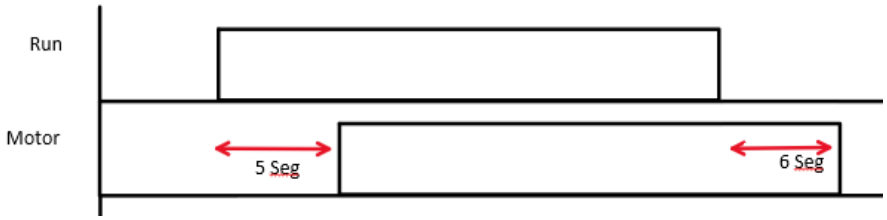

Figure 28: Diagrama tiempo motor, arranque y parada con temporizador

Para resolver este ejemplo se necesitará 2 temporizadores.
Uno con retardo de 5 segundos y el otro con retardo de 6 segundos, a continuación, se muestra sus diagramas de tiempo.

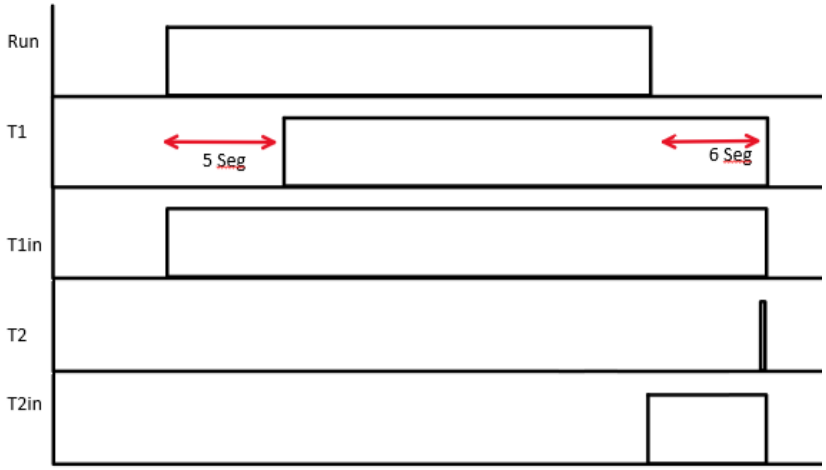

Figure 29: Diagrama tiempo temporizadores, arranque y parada con temporizadores

Al generar la función para el Temporizador 1 y 2 se encuentra incompatibilidad en el término mínimo “0” ya que este presenta 2 estados diferentes.

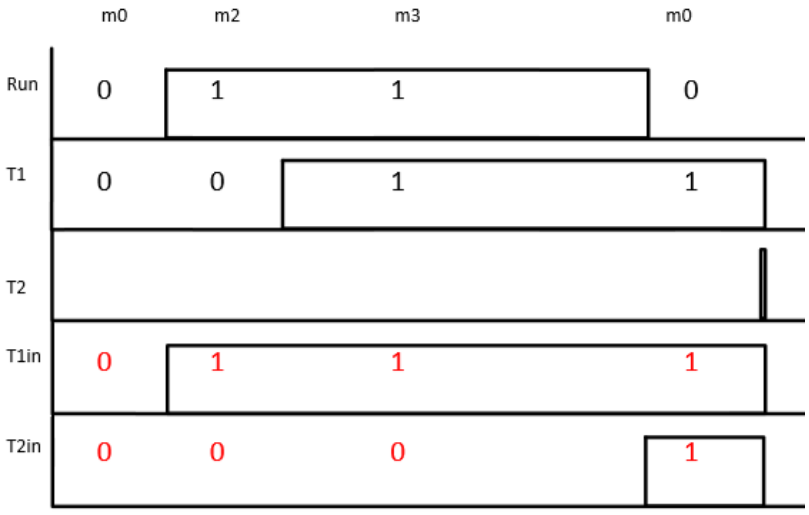

Figure 30: Obtención de términos mínimos desde Diagrama de tiempo

$$T1in = \sum(0,2,3) \prod(0)$$

$$T2in = \sum(0) \prod(0,2,3)$$

El problema radica en la variable Run, ya que solo tiene vigencia hasta 6 segundos antes de que culmine T1, para arreglar este inconveniente se plante una variable Aux que tenga toda la duración de tiempo.
Se observa que T1in se parce a un enclavador como el descrito anteriormente donde el "Start" es "Run" y "Stop" es "T2"
Esto deja la siguiente ecuación.

$$Aux = (Aux + Run)(T2)'$$
$$T1in = Aux$$

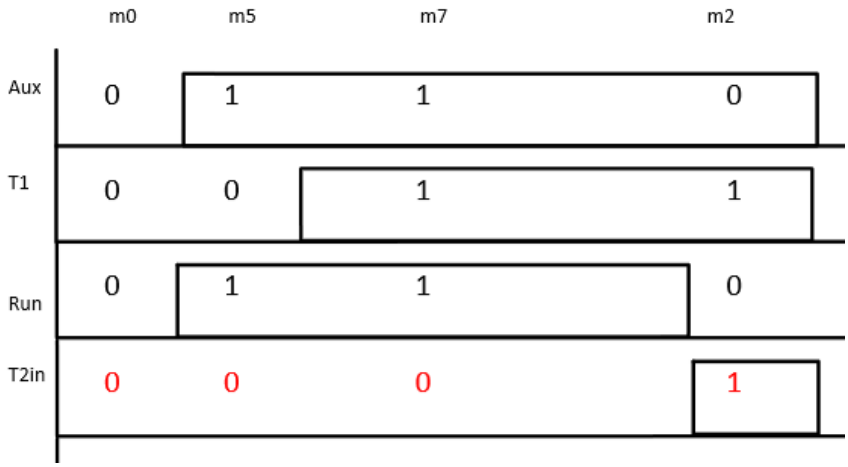

Figure 31: Obtención función T2in con variable auxiliar

$$T2in = \sum(2) \prod(0,5,7)$$

Simplificando y usando las condiciones de no importa

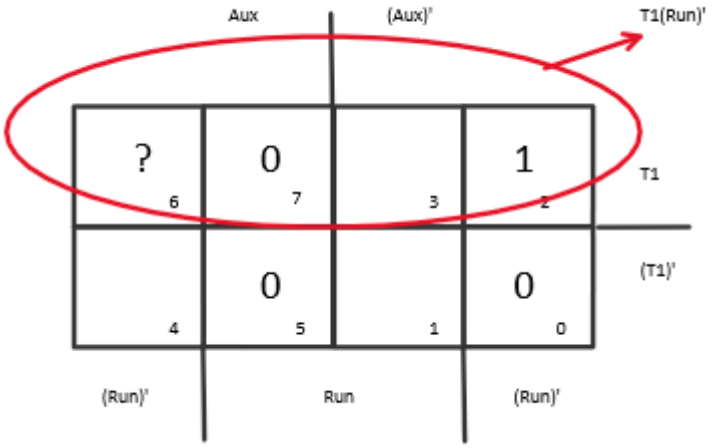

Figure 32: Simplificación T2in usando Karnaugh dependiente de variable auxiliar

Se tiene que.

$$T2in = (Run)'T1$$

De la gráfica anterior se calcula Motor

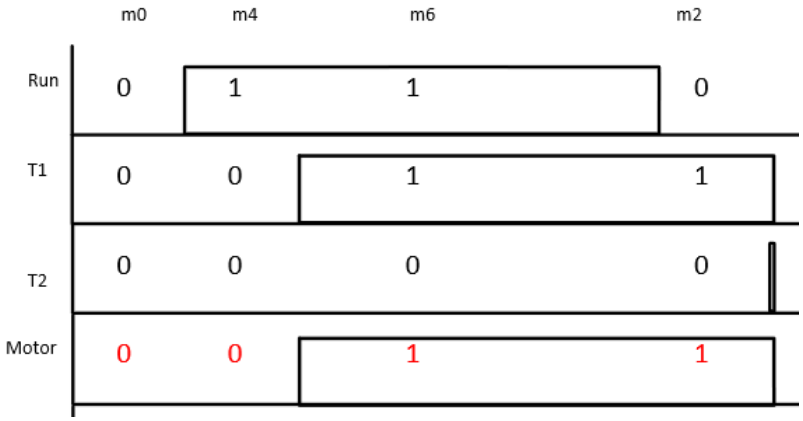

Figure 33: Obtención función motor con variable auxiliar

$$Motor = \sum(2,6) \prod(0,4)$$

Simplificando y usando las condiciones de no importa se tiene que.

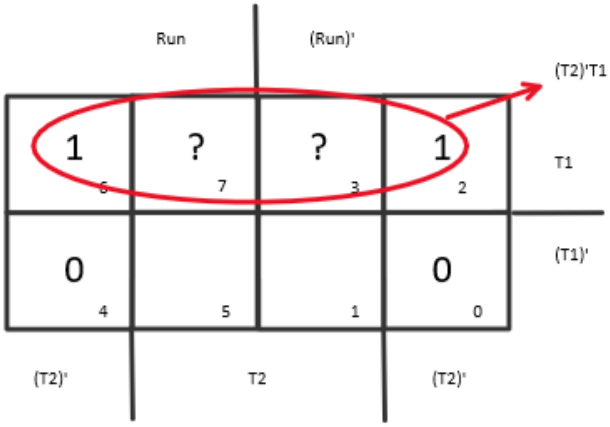

Figure 34: Simplificación función motor usando mapa de Karnaugh

$$Motor = T1$$

Por lo tanto, las ecuaciones finales son

$$Run = (Run + Start)(Stop)'$$
$$Aux = (Aux + Run)(T2)'$$
$$T1in = Aux$$
$$T2in = (Run)'T1$$
$$Motor = T1$$

La implementación es la siguiente.

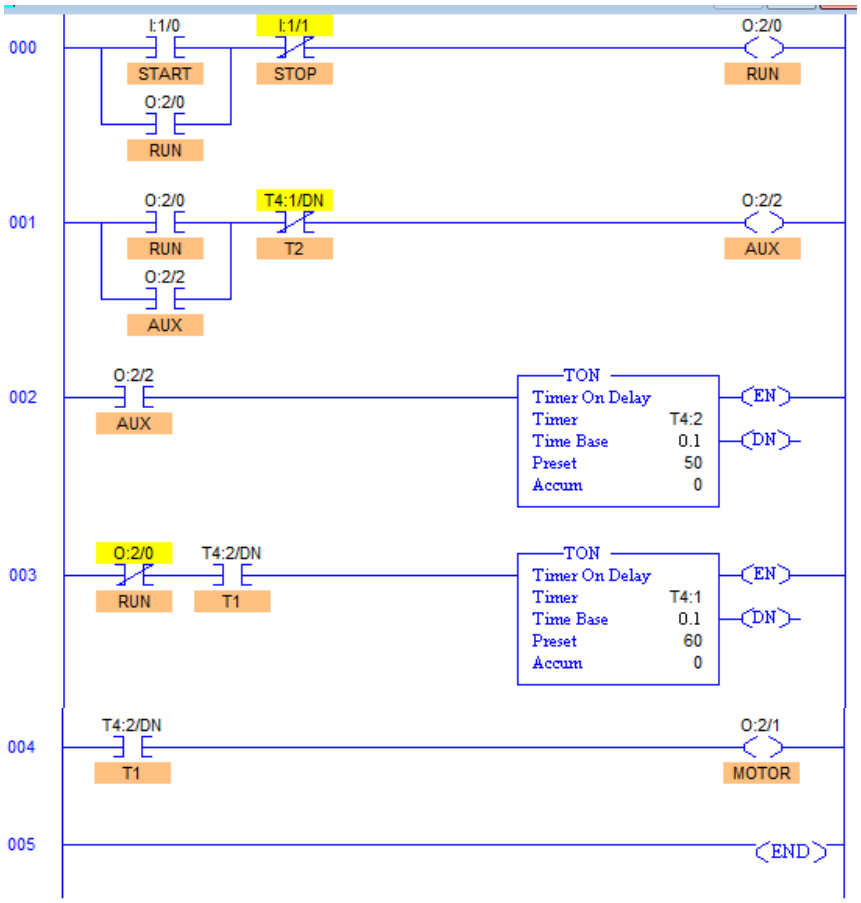

Figure 35: Implementación en lenguaje ladder usando variable auxiliar

Se puede realizar la misma solución sin el uso de variables auxiliares.
En el grafico se puede observar la dependencia de T1in y T2in de Run, T1 y T2.

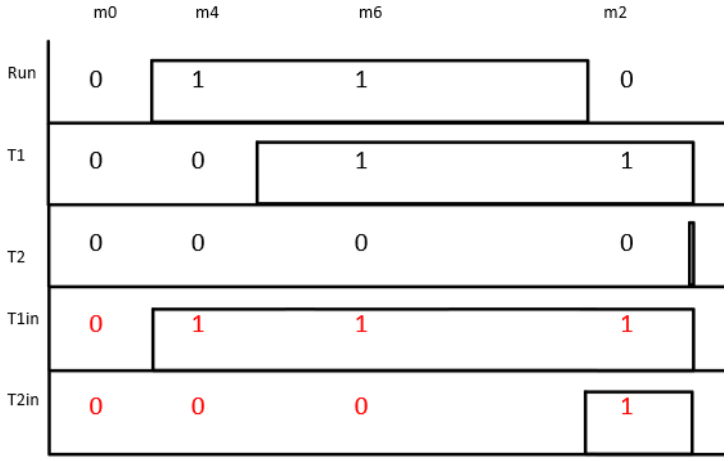

Figure 36: Obtención de términos mínimos para T1in y T2in usando diagramas de tiempo

$$T1in(Run, T1, T2) = \sum(2,4,6) \prod(0)$$

$$T2in(Run, T1, T2) = \sum(2) \prod(0,4,6)$$

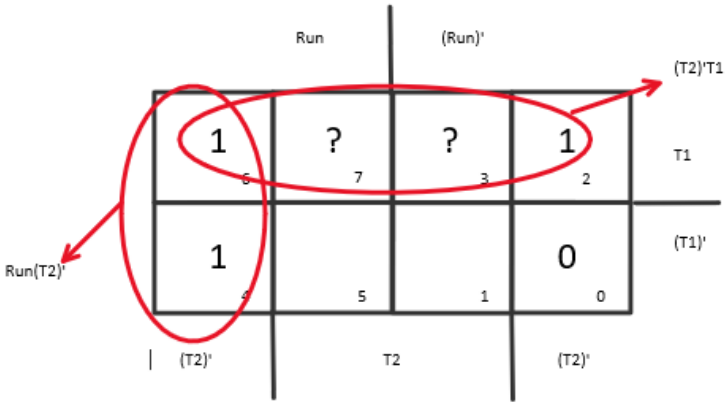

Figure 37: Simplificación de T1in usando mapa de Karnaugh

Los términos mínimos 3 y 7 no se toman porque cuando T2=1 el sistema se apaga, y este es el que hace números impares.

$$T1in = (T1 + Run)(T2)'$$

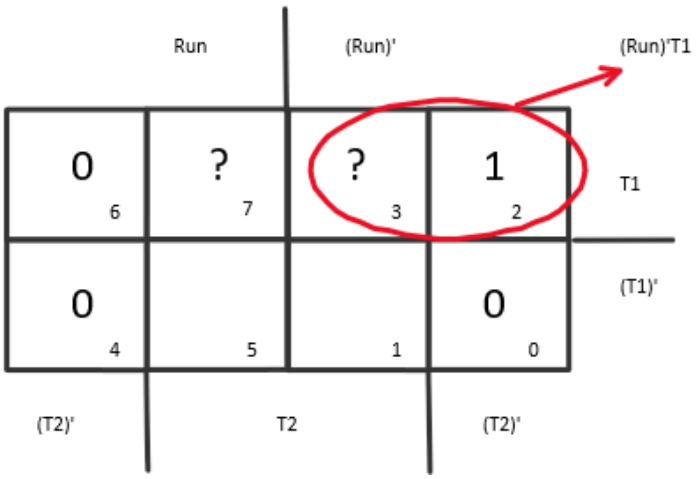

Figure 38: Simplificación T2in usando mapa de Karnaugh

$$T2in = (Run)'T1$$

Calculando la función para el motor se tiene que.

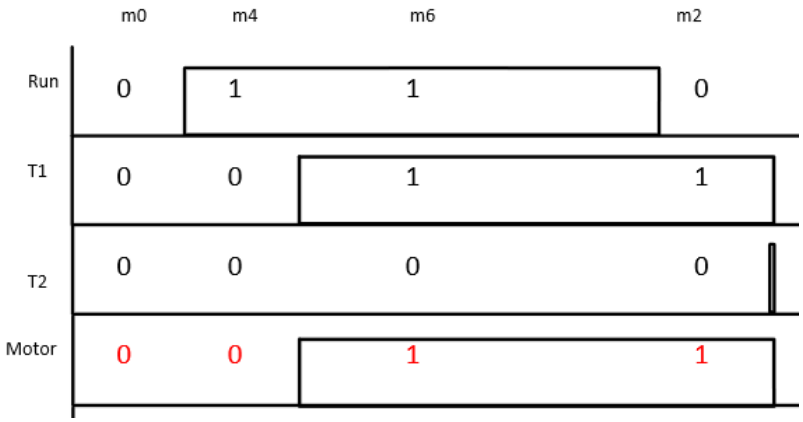

Figure 39: Obtención de términos mínimos para función motor a partir de diagrama de tiempo

$$Motor = \sum(2,6) \prod(0,4)$$

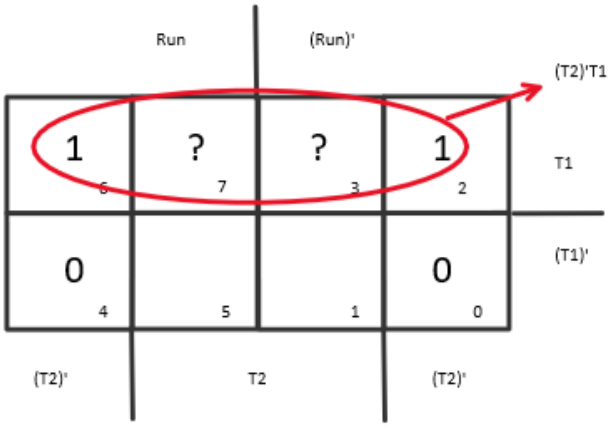

Figure 40: Simplificación función motor usando mapa de Karnaugh

$$Motor = T1$$

Las ecuaciones a implementar serán:

$$Run = (Run + Start)(Stop)'$$

$$T1in = T1 + Run(T2)'$$

$$T2in = (Run)'T1$$

$$Motor = T1$$

La implementación del diagrama de mando será.

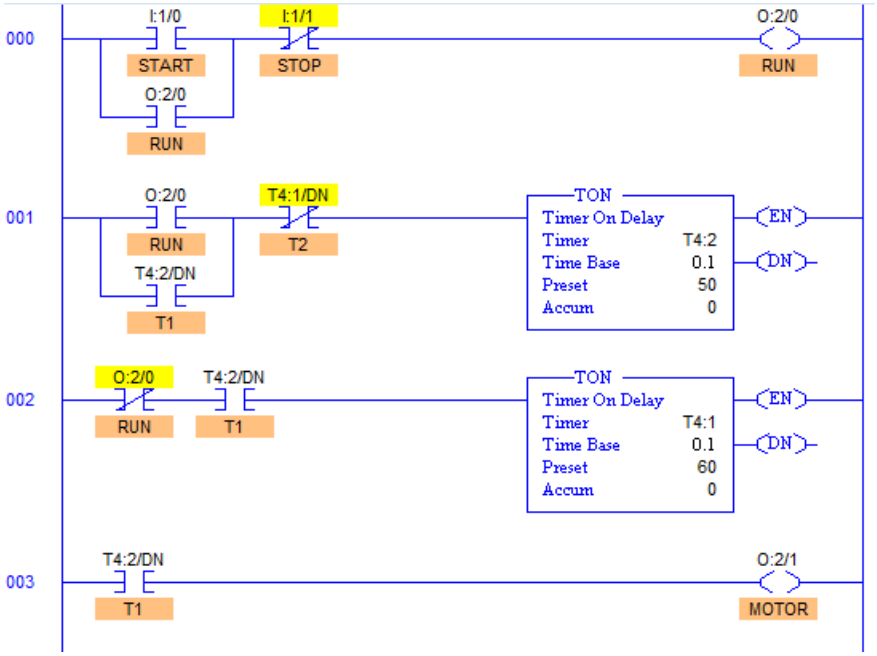

Figure 41: Implementación en lenguaje Ladder, motor con temporizador para arranque y desactivación

5.2.10. Metodología para la programación de PLC en lenguaje Ladder.

De lo descrito anteriormente se puede plantear un procedimiento para la realización de programas en lenguaje Ladder, basado en la herramienta de algebra Boolena. Los pasos a seguir son los siguientes:

A. Determinar los requerimientos del programa, cuantas variables serán entradas y cuantas salidas.
B. Si el sistema es temporizado, determinar el número de temporizadores a usar.
C. Realizar el diagrama de tiempos del sistema.
D. Determinar los términos mínimos, máximos y condiciones de no importa que caracterizan a la función.
E. Si los términos minimos y máximos son iguales, esto requiere el uso de una variable auxiliar para su desarrollo.
F. Simplificar las funciones que caracterizan el proceso
G. Realizar la implementación de la función en un Software de PLC para su validación.

5.2.11. Prueba de la metodología.

Para la prueba de la metodología se usará el siguiente Caso práctico.
Se tiene 2 fajas transportadoras de mineral, la primera faja recibe mineral de un Silo, luego está la transporta a la segunda faja que finalmente la deposita en un contenedor.
Al presionar Start la segunda faja debe de empezar a trabajar, debido a que, si contiene material empiece a liberarse de él, una vez libre de material que aproximadamente es 10 segundos debe de arrancar la primera faja y abrirse el Silo para que pueda transportar material.
Al presionar Stop el Silo debe de cerrarse y la primera faja dejara de funcionar al cabo de 10 segundos, tiempo necesario para que esta quede libre de material, y 10 segundos después debe de parar la segunda faja.
Aplicando la metodología tenemos que.

A. Requerimientos del programa.

a. Activar y desactivar el Proceso son Start y Stop, esto conlleva al uso de una variable interviniente llamada Run.
b. Controlar el Silo
c. Controlar la Faja1 y Faja2

B. Numero de Temporizadores.

a. Como el sistema esta temporizado para la primera faja se usarán 2 temporizadores, uno en el arranque y otro en el apagado
b. La segunda faja tiene un temporizado solo en el apagado, por lo tanto, se usara un temporizador
c. El total de temporizadores a usar será 3

C. Diagrama de tiempos del sistema.

Se presenta el diagrama de tiempos de los motores y el Silo.

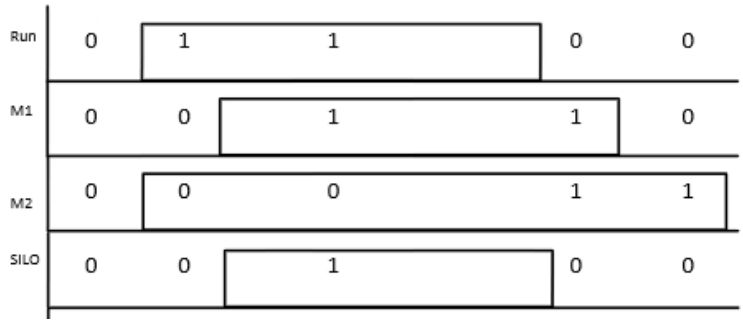

Figure 42: Diagrama de tiempo, control de motores con temporizadores y válvula de silo

Ahora se graficará el diagrama de tiempos de los tres temporizadores y Run.

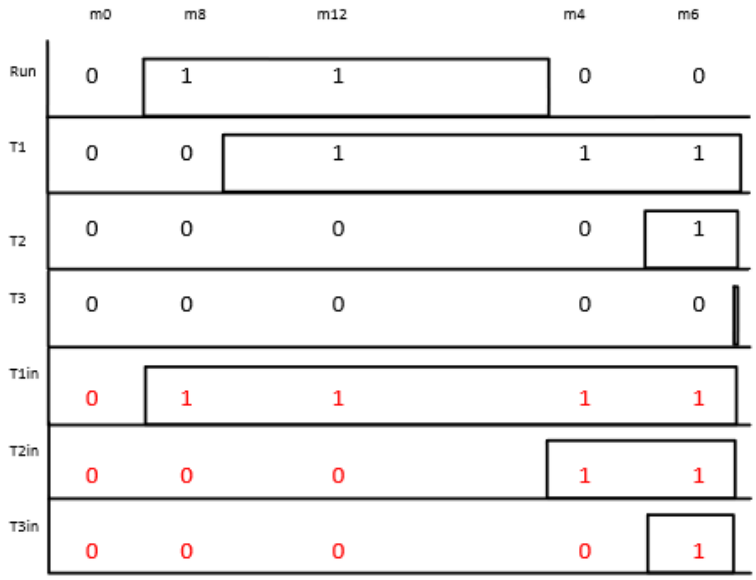

Figure 43: Diagrama de tiempo de temporizadores

Determinación del diagrama de tiempos de los motores en función de los temporizadores.

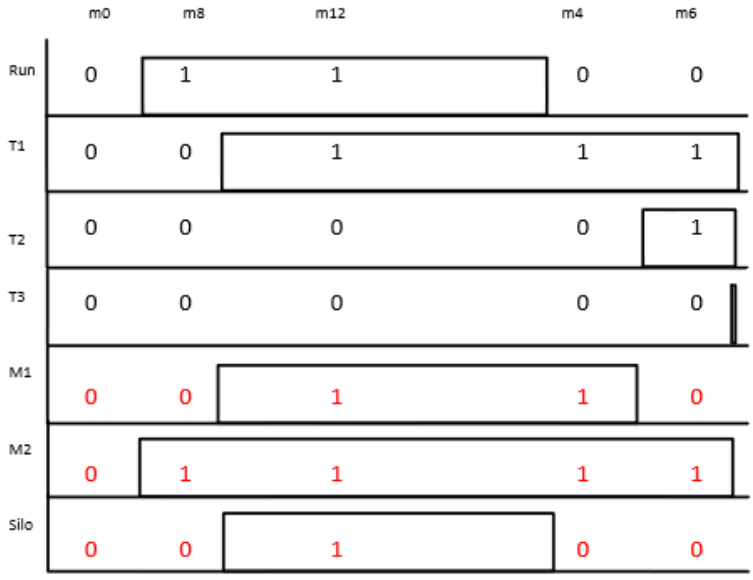

Figure 44: Diagrama de tiempo Motores de faja y válvula de silo

D. Determinación de los términos mínimos, máximos y condiciones de no me importa.

$$T1in = f(Run, T1, T2, T3) = \sum(4,6,8,12)$$
$$= \prod(0, impares)$$
$$T2in = f(Run, T1, T2, T3) = \sum(4,6)$$
$$= \prod(0,8,12, impares)$$

$$T3in = f(Run, T1, T2, T3) = \sum(6)$$
$$= \prod(0,4,8,12, impares)$$

$$M1 = f(Run, T1, T2, T3) = \sum(4,12)$$
$$= \prod(0,8,6, impares)$$
$$M2 = f(Run, T1, T2, T3) = \sum(4,6,8,12)$$
$$= \prod(0, impares)$$
$$Silo = f(Run, T1, T2, T3) = \sum(12)$$
$$= \prod(0,4,6,8, impares)$$

E. No existe términos mínimos y máximos iguales.

F. Simplificación de las funciones.

Calculando ***T1in***

$$T1in = f(Run, T1, T2, T3) = \sum(4,6,8,12)$$
$$= \prod(0, impares)$$

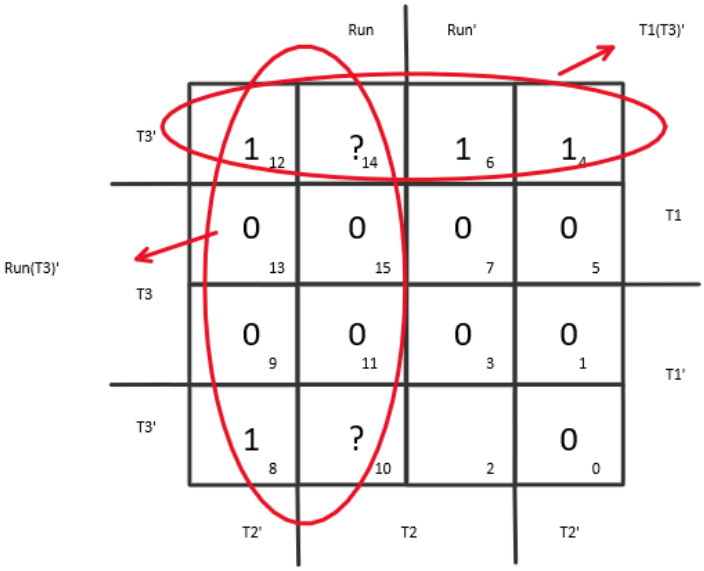

Figure 45: Simplificación función T1in

$$T1in = T1(T3)' + Run(T3)' = (T1 + Run)(T3)'$$

Calculando ***T2in***.

$$T2in = f(Run, T1, T2, T3) = \sum(4,6)$$
$$= \prod(0,8,12, impares)$$

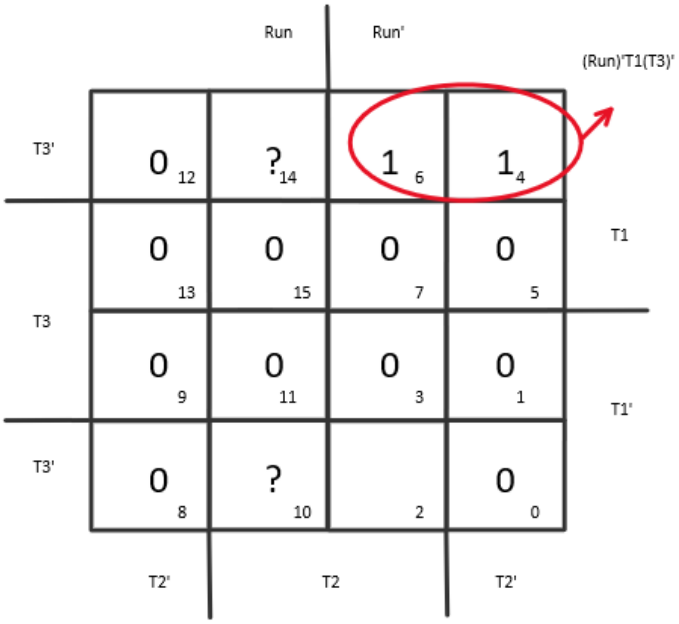

Figure 46: Simplificación T2in

$$T2in = (Run)'T1(T3)'$$

Calculando ***T3in***.

$$T3in = f(Run, T1, T2, T3) = \sum(6)$$
$$= \prod(0,4,8,12, impares)$$

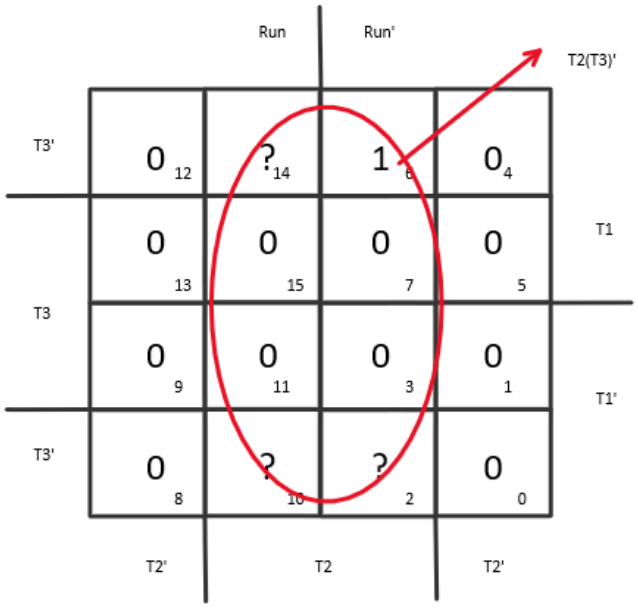

Figure 47: Simplificación T3in

$$T3in = T2(T3)'$$

Calculo de ***M1***.

$$M1 = f(Run, T1, T2, T3) = \sum(4,12)$$
$$= \prod(0,8,6, impares)$$

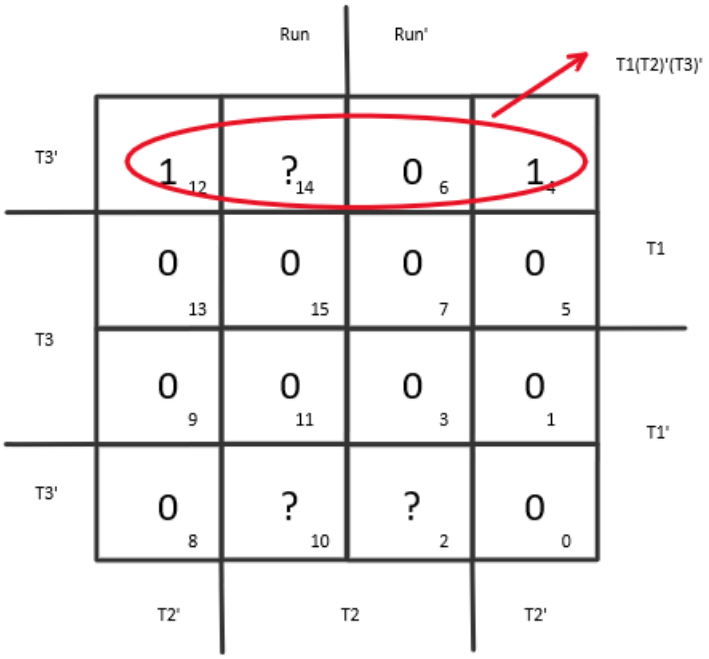

Figure 48: Simplificación Motor 1

$$M1 = T1(T2)'(T3)'$$

Calculo de ***M2***.

$$M2 = f(Run, T1, T2, T3) = \sum(4,6,8,12)$$
$$= \prod(0, impares)$$

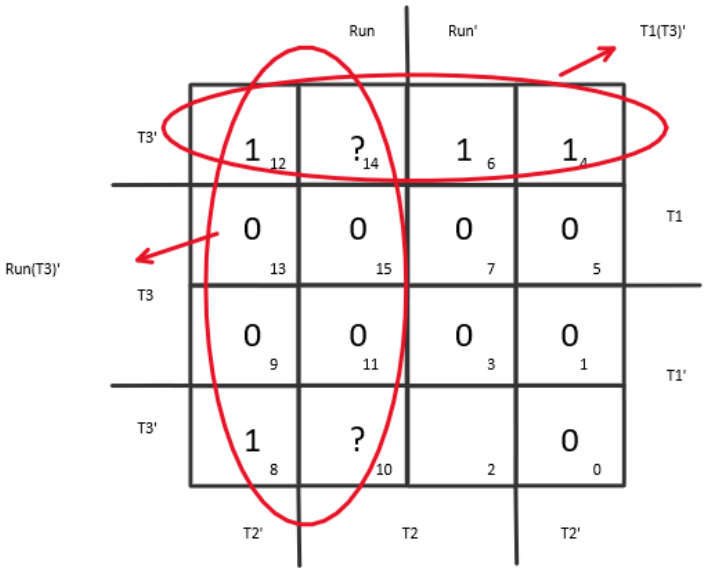

Figure 49: Simplificación Motor 2

$$M2 = T1(T3)' + Run(T3)' = (T1 + Run)(T3)'$$

$$M2 = T1in$$

Calculo de ***Silo***.

$$Silo = f(Run, T1, T2, T3) = \sum(12)$$
$$= \prod(0,4,6,8, impares)$$

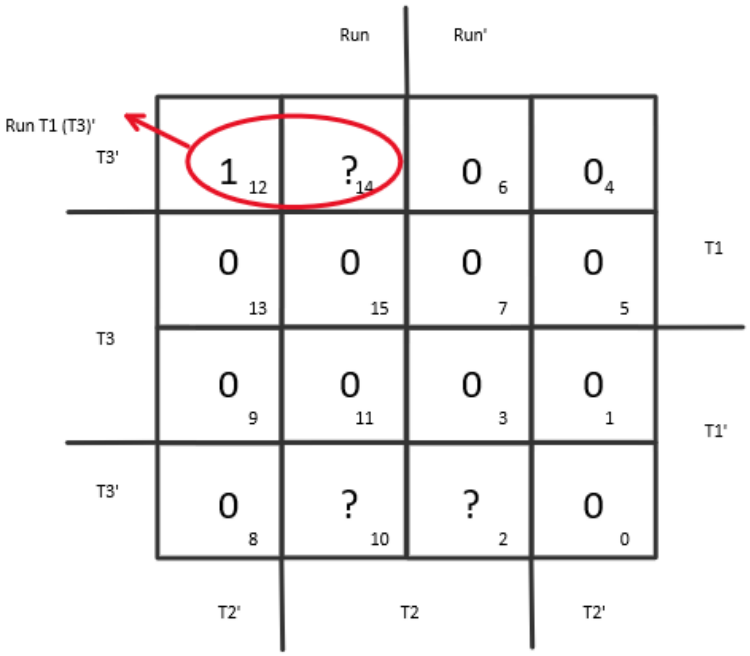

Figure 50: Simplificación función válvula de silo

$$Silo = Run\ T1\ (T3)'$$

G. Implementación del sistema.

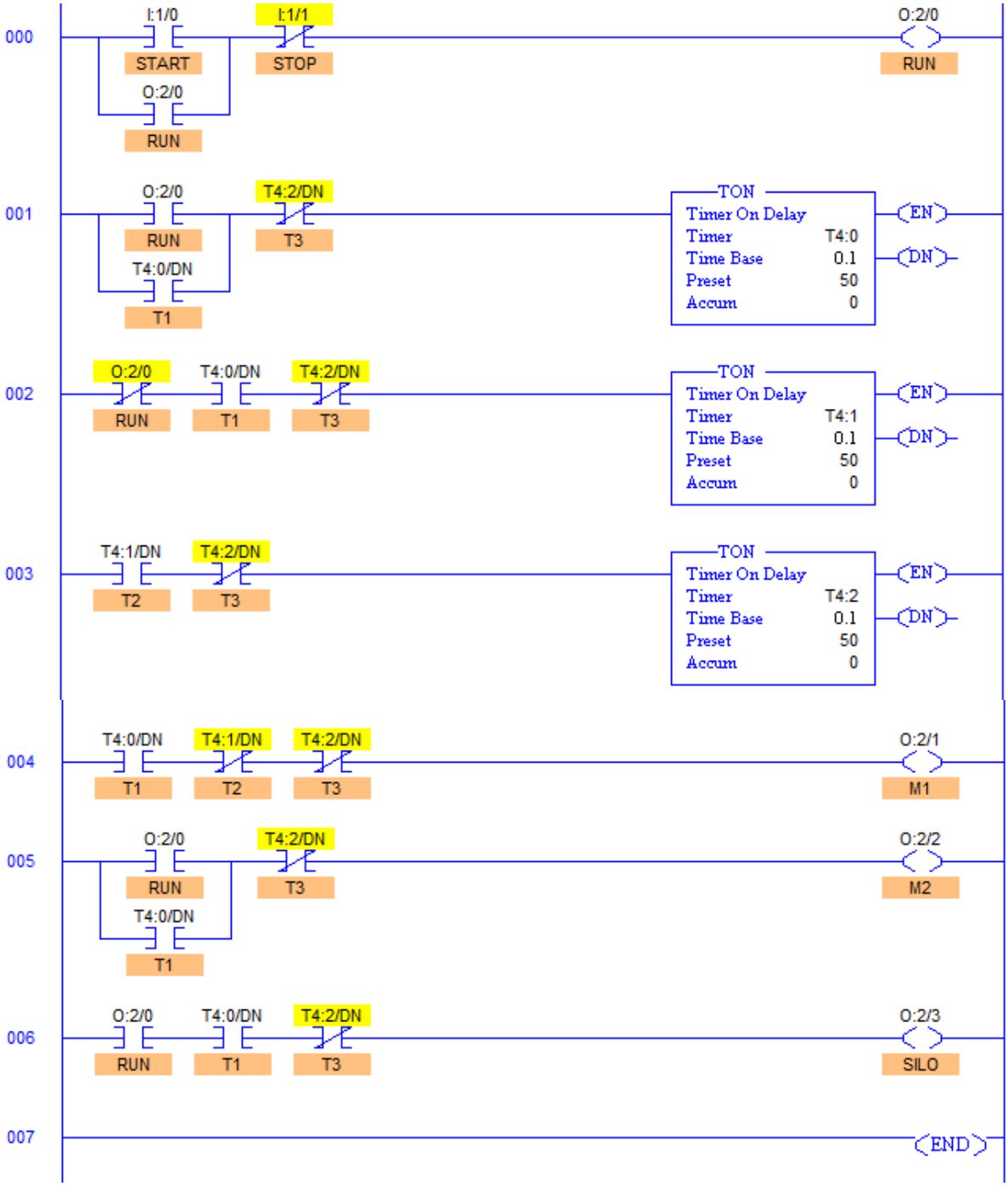

Figure 51: Implementación en Lenguaje Ladder control de motores con temporizador y válvula de silo

6. RESULTADOS

6.1. CIRCUITO SERIE

Cuando se tiene dos o más variables discretas en forma de producto, su equivalente circuital es una conexión en serie.

$$F(X,Y,Z) = XYZ$$

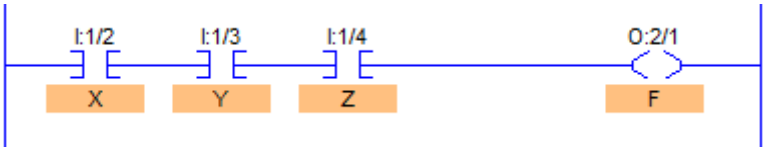

Figure 52: Implementación circuito serie

6.2. CIRCUITO PARALELO

Si las variables de una función booleana se suman circuitalmente indica que estas se encuentran en paralelo.

$$F(X,Y) = X + Y$$

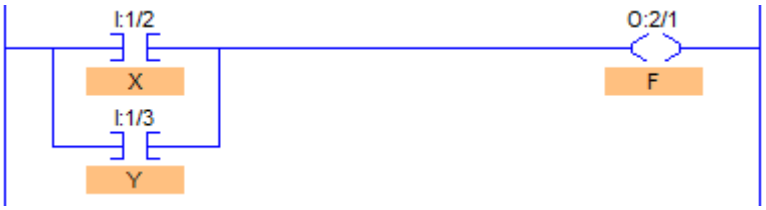

Figure 53: Implementación circuito paralelo

6.3. COMPLEMENTO

Cuando una variable se complementa indica que el interruptor esta normalmente cerrado.

$$F(X) = X'$$

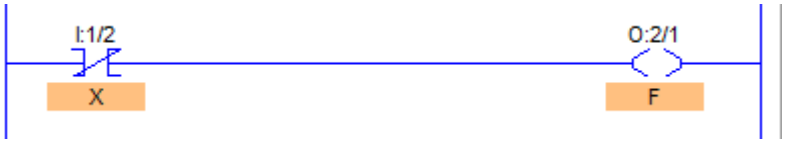

Figure 54: Implementación complemento

6.4. SIMPLIFICADO

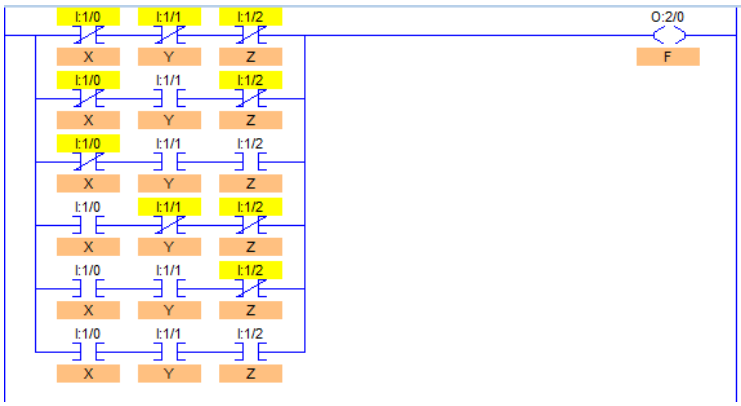

Figure 55: Implementación de esquema a simplificar

$$f(x,y,z) = x'y'z' + x'yz' + x'yz + xy'z' + xyz' + xyz$$

$$f(x,y,z) = \sum(0,2,3,4,6,7)$$

$$f(x,y) = y + z'$$

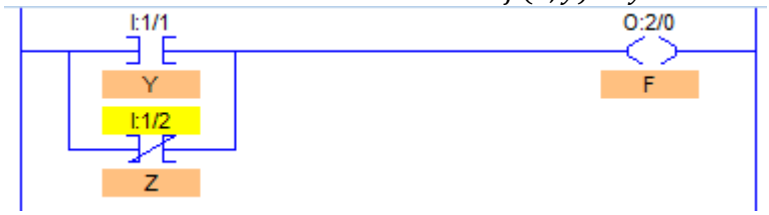

Figure 56: Implementación circuito simplificado

6.5. Modelamiento de procesos a partir de condiciones deseadas.

Se desea activar un motor con 2 interruptores, cada uno de ellos tiene la capacidad de activar o desactivar al motor independientemente del estado en que se encuentre el interruptor.
Para desarrollar este esquemático daremos nombres a cada interruptor.
La función booleana será.

$$f(x,y) = \sum(1,2) = \prod(0,3)$$

Termino	x	y	Motor
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

Tabla N° 24: Función motor para activación/desactivación de motor con dos interruptores

$$f(x,y) = x'y + xy'$$

La implementación será la siguiente.

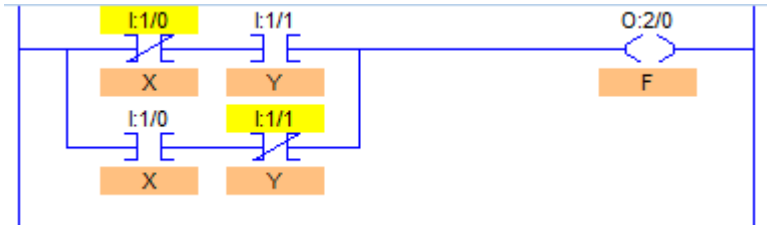

Figure 57: Implementación en Ladder control de motor con dos interruptores para activación/desactivación

6.6. Uso de variables de estado.

Desarrollar un programa que permita el enclavamiento de una salida usando 2 pulsadores, el primero activa la salida y el segundo al ser pulsado la desactiva.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
0	0	0	0	0
2	0	1	0	1
4	1	0	0	1
5	1	0	1	0
1	0	0	1	0
6	1	1	0	1
7	1	1	1	0
3	0	1	1	0

Tabla N° 25: enclavamiento de una salida usando 2 pulsadores

$$f(salida, Satrt, Stop)] = \sum(2,4,6) = \prod(0,1,3,5,7)$$

$$Salida = (Start + Salida)(Stop)'$$

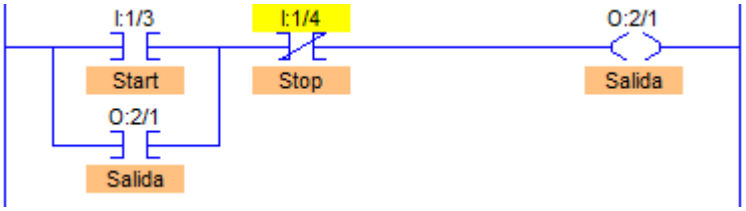

Figure 58: Implementación enclavamiento arranque / parada

Si al presionar Start y Stop a la vez, estos harán que se activara el motor. La función cambiara así.

Termino mínimo	Salida anterior	Start	Stop	Salida actual
7	1	1	1	1
3	0	1	1	1

Tabla N° 26: Presionar Start y Stop a la vez para que la salida se active

$$f(salida, Satrt, Stop)] = \sum(2,3,4,6,7) = \prod(0,1,5)$$

$$Salida = Start + Salida\ (Stop)'$$

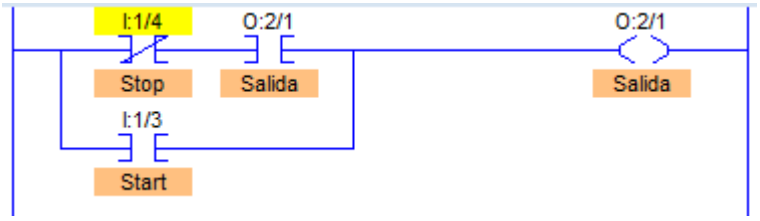

Figure 59: Implementación arranque parada segunda opción

6.7. Manejo de Diagramas de Tiempo.

Diseñar un diagrama de mando para arrancar 2 motores en secuencia usando un botón de Start y parar ambos con un botón de Stop.
El primer motor debe de arrancar cuando se presione Start
El segundo motor debe de arrancar 5 segundos después de arrancar el primero.

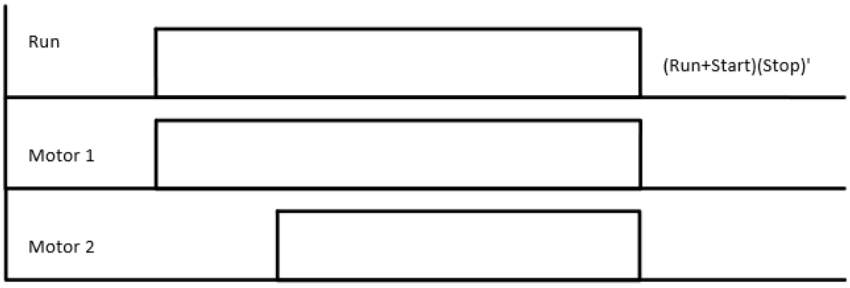

Figure 60: Diagrama de tiempo accionamiento de motores con temporizadores

Para arrancar el motor 2 después de 5 segundos se necesitará un temporizador

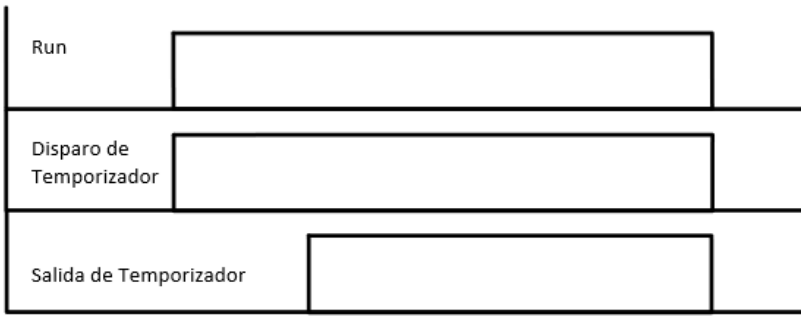

Figure 61: Diagrama de tiempo temporizador

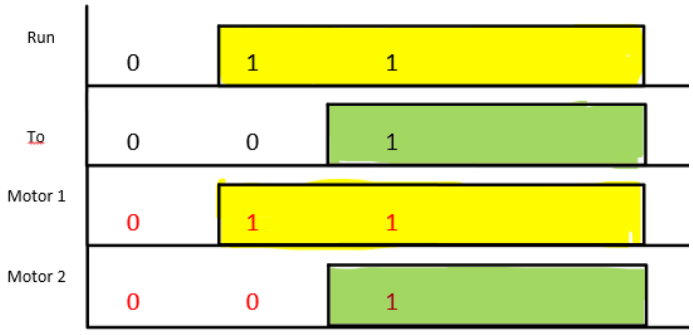

Figure 62: Obtención de términos mínimos a partir de diagrama de tiempo

$$Motor\ 1 = \sum(2,3) = \prod(0)$$

$$Motor\ 1 = Run$$

$$Motor\ 2 = \sum(3) = \prod(0,2)$$

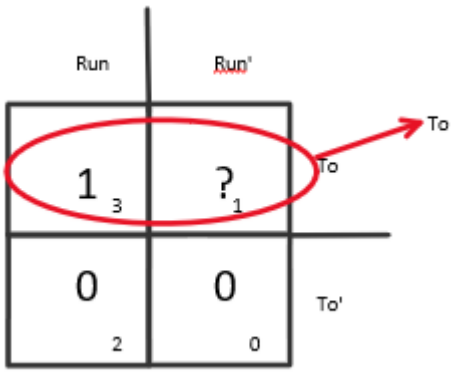

Figure 63: Simplificación Motor 2

$$Motor\ 2 = To$$

Este resultado se pudo observar de la gráfica anterior de tiempos.

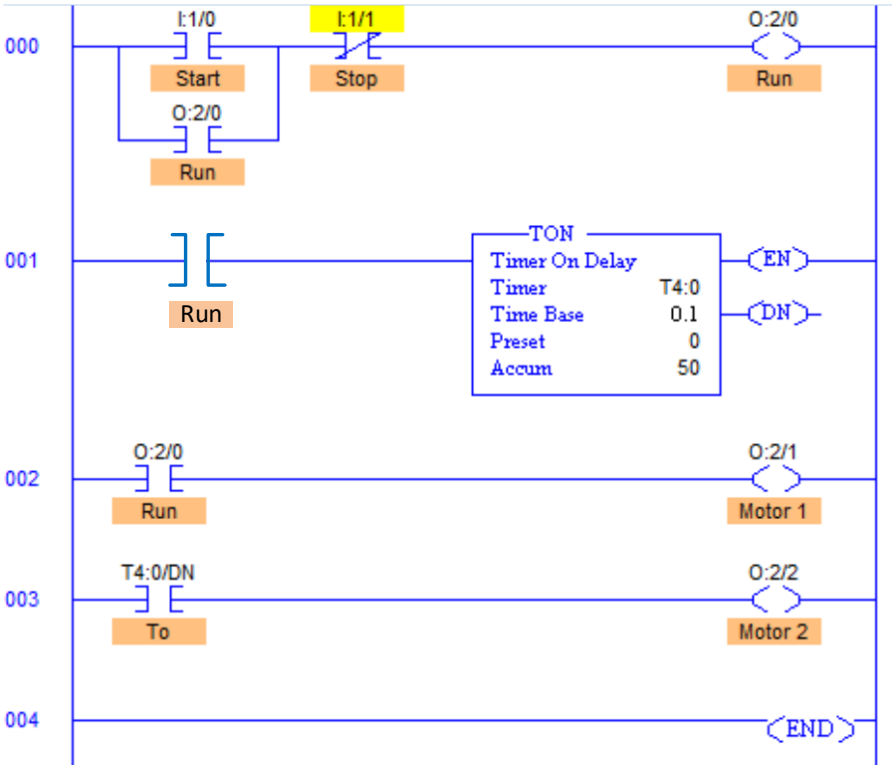

Figure 64: Implementación en Lenguaje Ladder arranque de motores con temporizadores

Uso de variables auxiliares.

Desarrollar un diagrama de mando para el siguiente sistema.
Un motor se activa 5 segundos después de presionar Start, y se apaga luego de 6 segundos de presionado Stop.

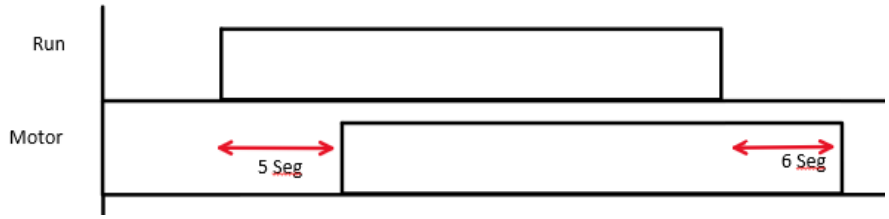

Figure 65: Diagrama de tiempo de arranque de motor con temporizador en arranque y parada

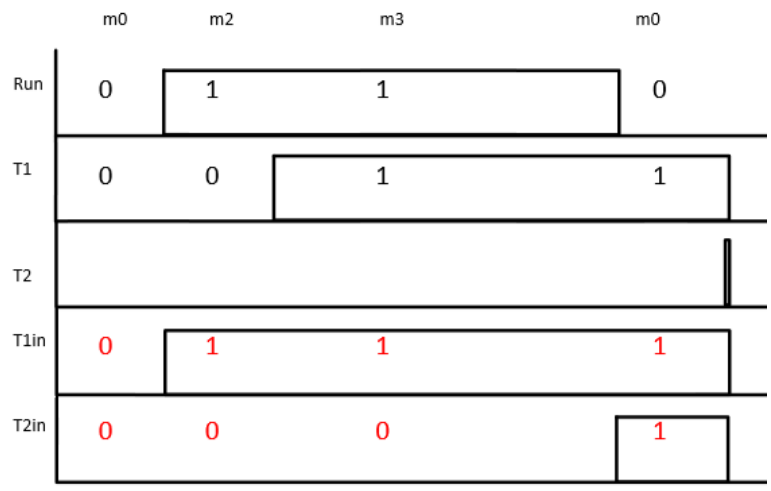

Figure 66: Obtención de términos mínimos para temporizadores

$$T1in = \sum(0,2,3)\prod(0)$$

$$T2in = \sum(0)\prod(0,2,3)$$

$$Aux = (Aux + Run)(T2)'$$
$$T1in = Aux$$

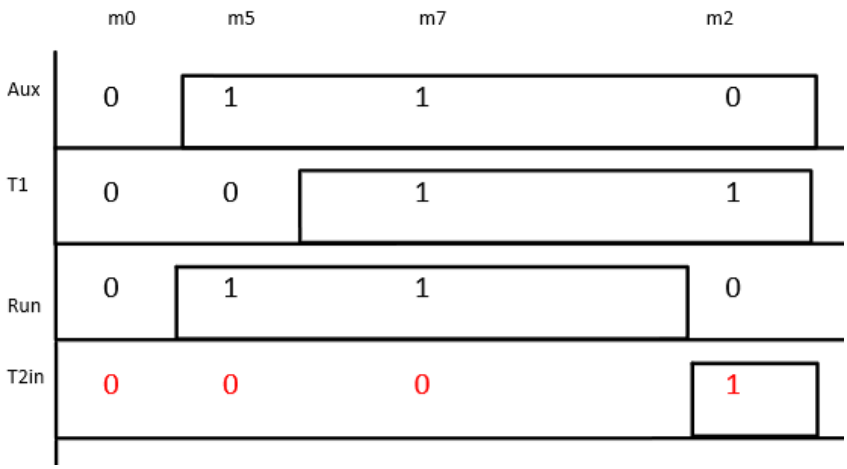

Figure 67: Obtención de términos mínimos para temporizador y motor usando variable auxiliar

$$T2in = \sum(2)\prod(0,5,7)$$

Se tiene que.

$$T2in = (Run)'T1$$

$$Motor = \sum(2,6)\prod(0,4)$$

Simplificando y usando las condiciones de no importa se tiene que.

$$Motor = T1$$

Por lo tanto, las ecuaciones finales son

$$Run = (Run + Start)(Stop)'$$
$$Aux = (Aux + Run)(T2)'$$
$$T1in = Aux$$
$$T2in = (Run)'T1$$
$$Motor = T1$$

La implementación es la siguiente.

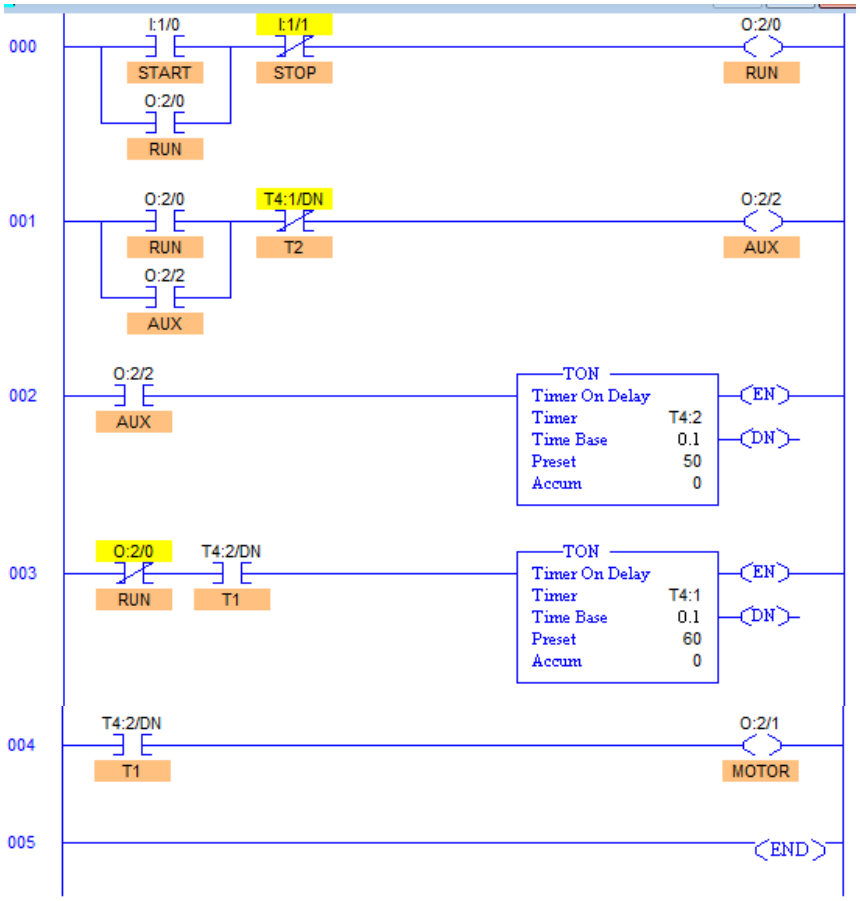

Figure 68: Implementación en lenguaje Ladder para motor con temporizadores en arranque y parada

6.8. Metodología para la programación de PLC en lenguaje Ladder.

De lo descrito anteriormente se puede plantear un procedimiento para la realización de programas en lenguaje Ladder, basado en la herramienta de algebra Boolena. Los pasos a seguir son los siguientes:

A. Determinar los requerimientos del programa, cuantas variables serán entradas y cuantas salidas.
B. Si el sistema es temporizado, determinar el número de temporizadores a usar.
C. Realizar el diagrama de tiempos del sistema.
D. Determinar los términos mínimos, máximos y condiciones de no importa que caracterizan a la función.

E. Si los términos mínimos y máximos son iguales, esto requiere el uso de una variable auxiliar para su desarrollo.
F. Simplificar las funciones que caracterizan el proceso
G. Realizar la implementación de la función en un Software de PLC para su validación.

6.9. Prueba de la metodología.

Para la prueba de la metodología se usará el siguiente Caso práctico.
Se tiene 2 fajas transportadoras de mineral, la primera faja recibe mineral de un Silo, luego está la transporta a la segunda faja que finalmente la deposita en un contenedor.
Al presionar Start la segunda faja debe de empezar a trabajar, debido a que, si contiene material empiece a liberarse de él, una vez libre de material que aproximadamente es 10 segundos debe de arrancar la primera faja y abrirse el Silo para que pueda transportar material.
Al presionar Stop el Silo debe de cerrarse y la primera faja dejara de funcionar al cabo de 10 segundos, tiempo necesario para que esta quede libre de material, y 10 segundos después debe de parar la segunda faja.
Aplicando la metodología tenemos que.

A. Requerimientos del programa.
   a. Activar y desactivar el Proceso son Start y Stop, esto conlleva al uso de una variable interviniente llamada Run.
   b. Controlar el Silo
   c. Controlar la Faja1 y Faja2

B. Numero de Temporizadores.
   a. El total de temporizadores a usar sera 3

C. Diagrama de tiempos del sistema.
   Se presenta el diagrama de tiempos de los motores y el Silo.

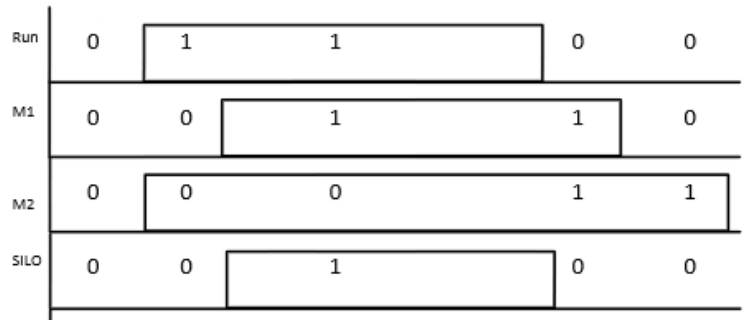

Figure 69: Diagrama de tiempo para control de motores de fajas y válvula de silo

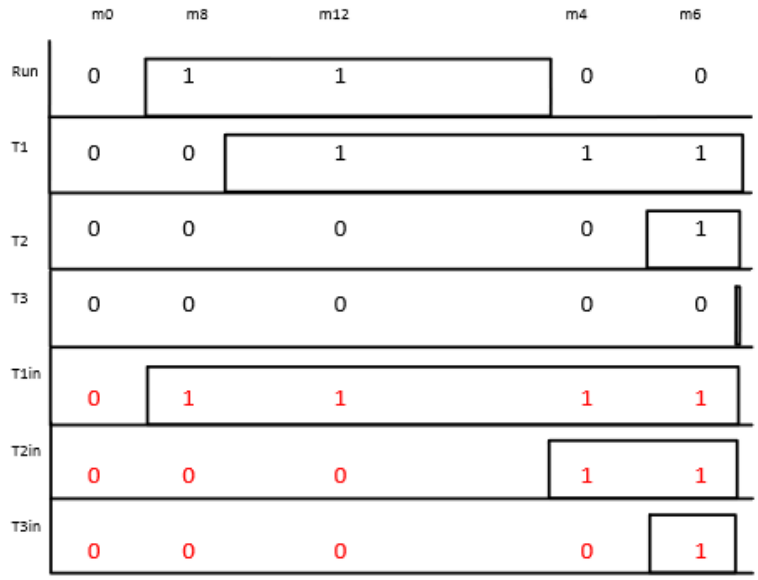

Figure 70: Obtención de términos mínimos para temporizadores

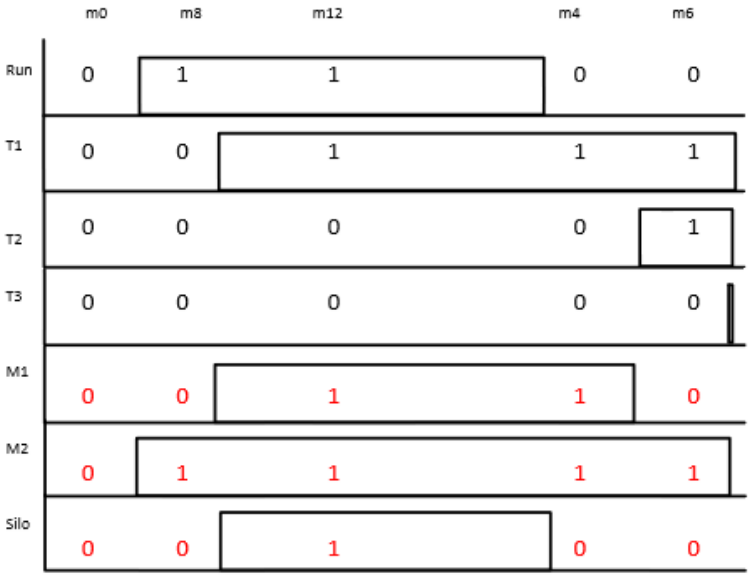

Figure 71: Obtención de Términos mínimos para motores de fajas y válvula de silo

D. Determinación de los términos mínimos, máximos y condiciones de no me importa.

$$T1in = f(Run, T1, T2, T3) = \sum(4,6,8,12)$$
$$= \prod(0, impares)$$
$$T2in = f(Run, T1, T2, T3) = \sum(4,6)$$
$$= \prod(0,8,12, impares)$$

$$T3in = f(Run, T1, T2, T3) = \sum(6) = \prod(0,4,8,12, impares)$$

$$M1 = f(Run, T1, T2, T3) = \sum(4,12) = \prod(0,8,6, impares)$$

$$M2 = f(Run, T1, T2, T3) = \sum(4,6,8,12) = \prod(0, impares)$$

$$Silo = f(Run, T1, T2, T3) = \sum(12) = \prod(0,4,6,8, impares)$$

E. No existe términos mínimos y máximos iguales.

F. Simplificación de las funciones.

Calculo de ***T1in***

$$T1in = f(Run, T1, T2, T3) = \sum(4,6,8,12) = \prod(0, impares)$$

$$T1in = T1(T3)' + Run(T3)' = (T1 + Run)(T3)'$$

Calculo ***T2in***.

$$T2in = f(Run, T1, T2, T3) = \sum(4,6) = \prod(0,8,12, impares)$$

$$T2in = (Run)'T1(T3)'$$

Calculo ***T3in***.

$$T3in = f(Run, T1, T2, T3) = \sum(6) = \prod(0,4,8,12, impares)$$

$$T3in = T2(T3)'$$

Calculo de ***M1***.

$$M1 = f(Run, T1, T2, T3) = \sum(4,12) = \prod(0,8,6, impares)$$

$$M1 = T1(T2)'(T3)'$$

Calculo de ***M2***.

$$M2 = f(Run, T1, T2, T3) = \sum(4,6,8,12) = \prod(0, impares)$$

$$M2 = T1(T3)' + Run(T3)' = (T1 + Run)(T3)'$$
$$M2 = T1in$$

Calculo de ***Silo***.

$$Silo = f(Run, T1, T2, T3) = \sum(12)$$
$$= \prod(0,4,6,8, impares)$$

$$Silo = Run\ T1\ (T3)'$$

G. Implementación del sistema.

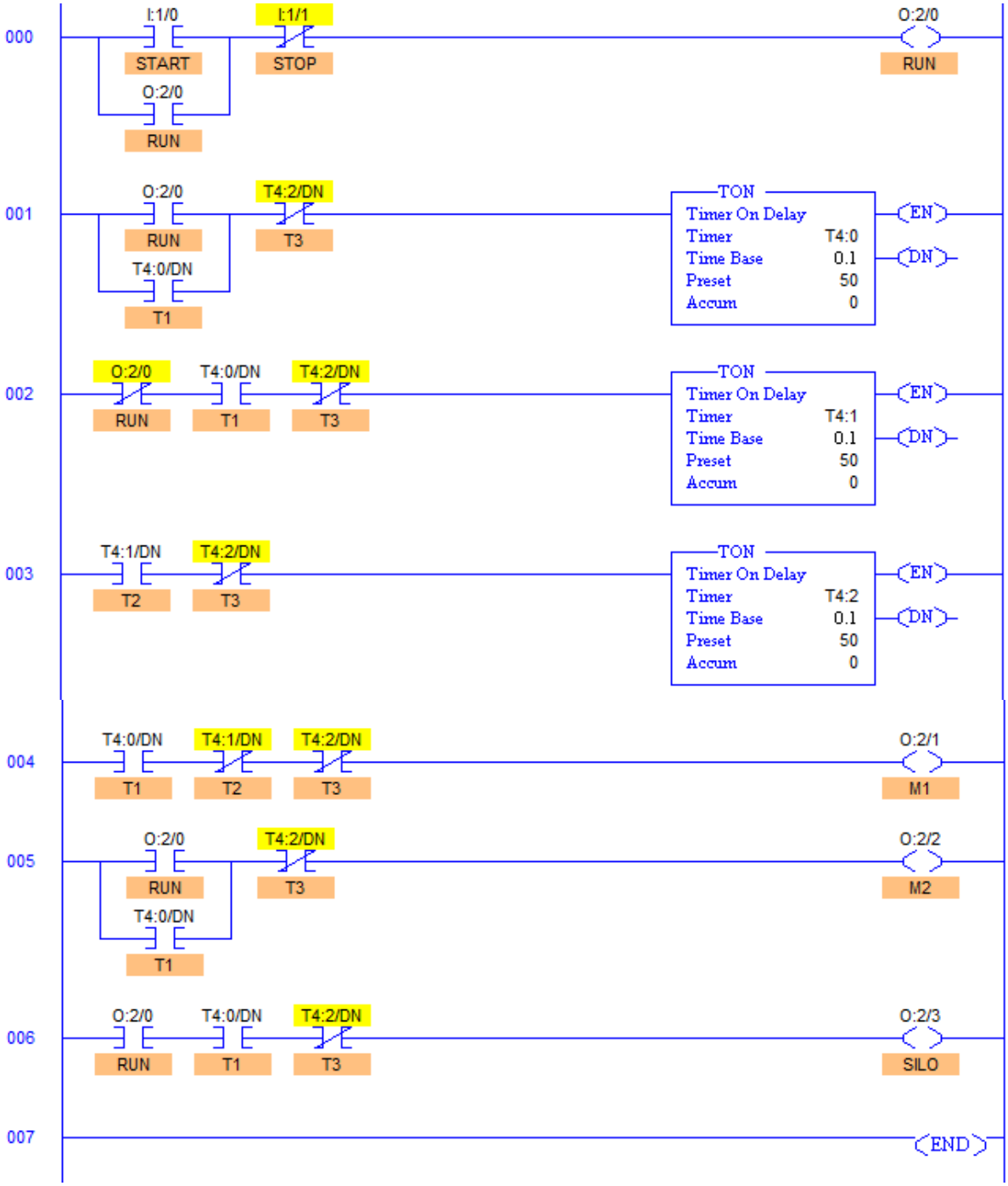

Figure 72: Implementación en Lenguaje Ladder para control de motores con temporizadores y apertura cierre de válvula de silo

7. CONCLUSIONES Y RECOMENDACIONES.

7.1. CONCLUSIONES.

- Se determinó la herramienta matemática que expresa el comportamiento de los contactos y esta es el Algebra booleana. Esto se evidencia en los puntos **5.2.1** hasta **5.2.5**
- Las técnicas que permiten la simplificación de funciones discretas son el álgebra de Boole y los mapas de Karnaugh, este último es el que se adoptó por ser el más práctico, tal como se evidencia en el Punto **5.2.5.**
- Se elaboró el algoritmo que permite la elaboración de programas en Ladder para PLC, esto se evidencia en el punto **5.2.10**
- Se realizó la prueba de la metodología al realizar un programa para controlar 2 fajas transportadoras de mineral, como se evidencia en el punto **5.2.11**.

7.2. RECOMENDACIONES.

- Concluida la tesis se considera investigar sobre una metodología para elaboración de algoritmos de programación de PLC modelando sistemas analógicos
- Al formular las ecuaciones que representan el sistema tener en cuenta que los términos mínimos y máximos deben de ser diferentes.
- Se debe tener en cuenta las condiciones de no importa para la simplificación de funciones
- Usar las ecuaciones de estado para generar variables auxiliares.

8. REFERENCIAS BIBLIOGRÁFICAS.

- Rolf,D. (1 de Mayo del 2018).“Introduccion a la programacion de controladores logicos (PLC)” .Recuperado de https://upload.wikimedia.org/wikipedia/ commons/6/65/Programacion_de_controladores_logicos_%28PLC%29.pdf

- Moreno, M.(5 de Mayo del 2018) “controlador lógico programable (plc) - Micro Automacion”. Recuperado de http://www.microautomacion.com/ capacitacion/Manual061ControladorLgicoProgramablePLC.pdf

- Aparicio, C. (2008) “Estructuración de sistemas de control de eventos discretos en un plc aplicado a procesos híbridos”, Mexico: Instituto Tecnológico y de Estudios Superiores de Monterrey

- Boscán, L. (2010) **"Diseño de un sistema de control mediante PLC para las instalaciones de aire acondicionado central (agua helada) e iluminación de un edificio de laboratorios”.**Caracas: Universidad Central de Venezuela.
- Trejo, S., Tejada D. (2014). **“Diseño de un sistema automático e instrumentación para la planta de almacenamiento y despacho de petróleo de la empresa Olympic Perú-Piura**”. Trujillo, Peru. Universidad Privada Antenor Orrego,
- Vega, A. (206).***”Diseño de un Circuito Logico para Funtores Logicos”***. Mexico. Universidad Naciona Autonoma de Mexico.