PROGRAMACIÓN E IMPLEMENTACIÓN DE UNA MESA DE DIBUJO DE CONTROL NUMÉRICO

VIVIAN SUSANA BLANQUICET RANGEL
VLADIMIR REMOLINA LÓPEZ

UNIVERSIDAD PONTIFICIA BOLIVARIANA
FACULTAD DE INGENIERÍA ELECTRÓNICA
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN
SECCIONAL BUCARAMANGA
2013

PROGRAMACIÓN E IMPLEMENTACIÓN DE UNA MESA DE DIBUJO DE CONTROL NUMÉRICO

VIVIAN SUSANA BLANQUICET RANGEL
VLADIMIR REMOLINA LÓPEZ

TRABAJO DE GRADO

MSC. JUAN CARLOS VILLAMIZAR
DIRECTOR DE PROYECTO

UNIVERSIDAD PONTIFICIA BOLIVARIANA
FACULTAD DE INGENIERÍA ELECTRÓNICA
ESCUELA DE INGENIERÍA Y ADMINISTRACIÓN
SECCIONAL BUCARAMANGA
2013

Nota de Aceptación

Este trabajo ha sido revisado y analizado. Se ha determinado que reúne los requisitos de elaboración y presentación exigidos, por tanto se notifica su aceptación.

_____

Firma de presidente de jurado

_____

Firma de jurado

_____

Firma de jurado

Bucaramanga, Noviembre 2013

DEDICATORIA

A mis padres que han sido fuente de inspiración y apoyo constante, a mis hermanas, por ser mi equilibrio, y mis mejores amigas, a cada persona que de alguna forma me acompaño, guió y ayudó a conseguir este logro, a todos ustedes gracias por su paciencia y comprensión.

VIVIAN SUSANA BLANQUICET RANGEL

A mi familia, especialmente a mis padres, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento, por ustedes es que he podido lograr mis metas, gracias por ser mí guía y ejemplo.

VLADIMIR REMOLINA LÓPEZ

AGRADECIMIENTOS

A Msc. Juan Carlos Villamizar Rincón por su apoyo, disponibilidad, paciencia y confianza en la realización de este trabajo de grado, por su capacidad para guiar nuestras ideas, no cabe duda que ha sido un aporte invaluable no solamente para el desarrollo de esta tesis sino también para nuestra formación como profesionales íntegros, su participación ha enriquecido el trabajo realizado y además ha significado el surgimiento de una sólida amistad.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

CÓDIGOS G: Lenguaje de programación usado en control numérico, el cual posee múltiples implementaciones. Usado principalmente en automatización, forma parte de la ingeniería asistida por computadora.

ETHERNET: Estándar de transmisión de datos para redes de área local que se basa en el principio en donde todos los equipos en una red Ethernet están conectados a la misma línea de comunicación compuesta por cables cilíndricos.

PLC TWIDO: Dispositivo digital electrónico con una memoria programable para el almacenamiento de instrucciones, permitiendo la implementación de funciones específicas como son: lógicas, secuenciales, temporizadas, de conteo y aritméticas; con el objeto de controlar máquinas y procesos. TWIDO es una marca de PLC de gran avance tecnológico y de fácil programación e interacción con el usuario.

PROTOCOLO MODBUS TCP/IP: Es un protocolo de comunicación diseñado para permitir a equipo industriales, tales como, controladores lógicos programables (PLCs), computadores, motores, sensores, y otros tipos de dispositivos físicos de entrada/salida comunicarse sobre una red. La especificación Modbus/TCP define un estándar interoperable en el campo de la automatización industrial, el cual es simple de implementar para cualquier dispositivo que soporta sockets9 TCP/IP.

REDES: conjunto de computadoras y otros equipos interconectados, que comparten información, recursos y servicios. Puede a su vez dividirse en diversas categorías, según su alcance (red de área local oLAN, red de área metropolitana o MAN, red de área amplia o WAN, etc.), su método de conexión (por cable coaxial, fibra óptica, radio, microondas, infrarrojos) o su relación funcional (cliente-servidor, persona a persona), entre otras.

RUNG: Nombre de la línea o escalón usado en la programación del PLC TWIDO en el entorno de programación TwidoSuite.

SOFTWARE: Equipamiento lógico de un sistema informático que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware.

TORNO DE CONTROL NUMÉRICO: Máquina herramienta del tipo torno que se utiliza para mecanizar piezas de revolución mediante software de computadora que utiliza datos alfa-numéricos, siguiendo los ejes cartesianos X,Y,Z. se utiliza para para producir en cantidades y con precisión porque la computadora que lleva incorporada controla la ejecución de la pieza.

TWIDOSUITE: Software usado para la programación del PLC TWIDO.

VISUAL BASIC: Lenguaje de programación dirigido por eventos, desde su primera versión utiliza un ambiente de desarrollo completamente gráfico que facilita la creación de interfaces gráficas y en cierta medida, también la programación misma.

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: PROGRAMACIÓN E IMPLEMENTACIÓN DE UNA MESA DE DIBUJO DE CONTROL NUMÉRICO

AUTOR(ES): VIVIAN SUSANA BLANQUICET RANGEL
VLADIMIR REMOLINA LÓPEZ

FACULTAD: Facultad de Ingeniería Electrónica

DIRECTOR(A): JUAN CARLOS VILLAMIZAR RINCÓN

RESUMEN

Se crea software de fácil manejo para máquinas de control numérico implementado en una mesa de dibujo, la mesa de dibujo se diseñó especialmente para seguir las coordenadas previamente introducidas en el software. El software se desarrolla en el entorno de programación de Visual Basic, en él, se ejecuta una interfaz gráfica para que el usuario, por medio de códigos G, realice y depure la figura que luego será plasmada en la mesa de dibujo. La mesa de dibujo es dirigida por medio de un controlador lógico programable PLC, la marca elegida es TWIDO de TELEMECANIQUE debido a su funcionalidad y avance tecnológico hasta el momento. El control de la mesa consiste básicamente en el movimiento de 2 (dos) motores paso a paso que se mueven en un eje X y un eje Z y un motor de corriente continua que es energizado para controlar el lápiz encargado de recrear la figura del software el cual corresponde al eje Y. Se logra una comunicación sencilla a través de una red Ethernet mediante protocolo Modbus TCP/IP, ya que el PLC TWIDO de TELEMECANIQUE con referencia TWDLCAE40DRF posee un puerto RJ45 Ethernet integrado. La comunicación se realiza entre el PLC y el entorno de programación de Visual Basic. Con el software creado en Visual Basic y la comunicación establecida entre este entorno de programación y el PLC, se garantiza que la mesa de dibujo será capaz de reproducir cualquier figura introducida en el software ya sea equivalente a una línea recta o a una curva.

PALABRAS CLAVES: Modbus, TCP/IP, Torno control numérico, Visual Basic, PLC TWIDO TELEMECANIQUE

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: PROGRAMMING AND IMPLEMENTATION OF A DRAWING TABLE FOR NUMERICAL CONTROL MACHINES.

AUTHOR(S): VIVIAN SUSANA BLANQUICET RANGEL
VLADIMIR REMOLINA LÓPEZ

FACULTY: Facultad de Ingeniería Electrónica

DIRECTOR: JUAN CARLOS VILLAMIZAR RINCON

ABSTRACT

It creates user-friendly software for CNC machines deployed on a drafting table, drawing table is specially designed to follow the coordinates previously introduced in the software. Software is developed in the programming environment of Visual Basic, it is a graphical interface for the user, through codes G, complete and debug figure which is then reflected in the drawing board. The drawing board is led by a programmable logic controller PLC, the brand of choice is TWIDO of TELEMECANIQUE due to its functionality and technological progress so far.
Control table is basically the movement of two (2) step motors which move in an X-axis and Z-axis and the DC motor is energized to control the pen Figure manager software recreates which corresponds to the axis Y. Communication is achieved via a simple Ethernet network using Modbus TCP / IP, since TELEMECANIQUE PLC with reference TWIDO of TWDLCAE40DRF has an integrated Ethernet RJ45 port. Communication takes place between the PLC and the programming environment of Visual Basic. With software created in Visual Basic and established communication between the programming environment and the PLC ensures that the drawing board will be able to reproduce any figure entered in the software and is equivalent to a straight line or a curve.

KEYWORDS: Networks, Modbus, TCP/IP, CNC lathe, Visual Basic, PLC TWIDO TELEMECANIQUE

V° B° DIRECTOR DE TRABAJO DE GRADO

INTRODUCCION

Desde el principio el hombre siempre ha sentido la necesidad de crear y moldear herramientas para facilitar su vida y la de su sociedad. Este pensamiento ha jugado un papel muy importante en su propia evolución y en la evolución de la tecnología generando en la actualidad máquinas herramienta.

Las máquinas herramienta se utilizan para moldear y dar forma a objetos de diferentes materiales como metales, madera, etc., estas máquinas son muy usados en la industria. Estas mismas son de uso complejo al igual que su sistema de control, pero aun con estos inconvenientes ofrecen un aumento en la productividad de cualquier industria, ya sea grande, mediana o pequeña; por lo que han sido de gran aceptación en todo el mundo.

El principio de control actual para estas máquinas herramienta es el control numérico, lo que hace que sean máquinas más sencillas de implementar, más seguras para un operario y versátiles.

El control numérico es un sistema de automatizado de máquinas herramienta, tanto simples como complejas; tales como taladradoras, fresadoras, tornos, etc. Esta automatización se hace por medio de un software de programación por computador, reduciendo así el tiempo de trabajo y aumentando su simpleza y comodidad a la hora de su manipulación. A pesar que estas máquinas herramienta brindan tantos beneficios a los usuarios tienen el factor económico en contra pues son máquinas de alto costo y esto hace que no sean de fácil adquisición.

Por lo mencionado anteriormente, se ha creado un software de fácil manejo para máquinas de control numérico y ha sido implementado en una mesa de dibujo para máquinas de control numérico.

El software fue creado bajo el entorno de programación Visual Basic, lo que proporciona un fácil manejo del mismo, el control llevado a cabo en la mesa de dibujo implementada, se realiza por medio de un PLC generando así un avance tecnológico de importancia.

En el presente informe se encuentran descritos los elementos usados para el desarrollo de este trabajo de grado, esto incluye los motores paso a paso que controlan el movimiento de la estructura, el circuito de potencia para el control de los mismos motores paso a paso, el módulo de salidas transistorizadas adquirido para mejorar el rendimiento de los motores paso a paso, el controlador lógico programable PLC TWIDO TELEMECANIQUE, la interfaz gráfica desarrollada en Visual Basic y el protocolo de comunicación establecido para el funcionamiento completo de la mesa de dibujo, al igual que la descripción detallada de cada parte de la mesa de dibujo desarrollada.

Con el desarrollo de este trabajo de grado se espera que a partir de él y sus posteriores investigaciones, se produzca una reducción en los costos de una máquina de control numérico CNC usadas en la industria en general, al igual se desea que la mesa de dibujo pueda ser implementada a futuro y con ciertas mejoras, como parte de los laboratorios de automatización de la Universidad Pontificia Bolivariana.

OBJETIVOS

GENERAL

Desarrollar una mesa de dibujo de control numérico usando códigos G, controlada por medio de un PLC y computador personal.

ESPECIFÍCOS

- Desarrollar una interfaz gráfica para dibujar piezas usando códigos G.
- Desarrollar un programa en el PLC TWIDO para controlar motores paso a paso y hacer el trazado de líneas y círculos.
- Desarrollar un programa de comunicación entre el computador y el PLC usando comunicación Modbus sobre TCP/IP.
- Construir una mesa posicionadora que hará las funciones de la dibujadora, controlada por el software que se espera diseñar.

1. CONTROL NUMÉRICO

El control numérico es una forma de automatización programable; en la cual la máquina herramienta de procesado se controla a través de números, letras y otros símbolos, depositados en un medio de almacenamiento. Cuando la tarea en cuestión cambia, se cambia el programa de instrucciones.

El primer desarrollo en el área del control numérico se realizó en la década de 1940 basados en máquinas ya construidas y motores modificados.

Los caracteres más usados que se establecieron están regidos bajo la norma DIN 66024 y 66025 son, entre otros, los siguientes:

N que corresponde al número de bloque o secuencia. Esta letra va seguida de un número que corresponde a cada bloque diferente que es necesario programar. El número máximo de bloques que pueden programarse actualmente es de 9999.

X, Y, Z son las que se utilizan para señalar las cotas correspondientes a los ejes de coordenadas X, Y, Z de la máquina herramienta. En los tornos solo se utilizan las coordenadas X y Z. Dichas cotas se pueden programar en forma absoluta o relativa, es decir, con respecto al cero de la pieza o con respecto a la última cota respectivamente.

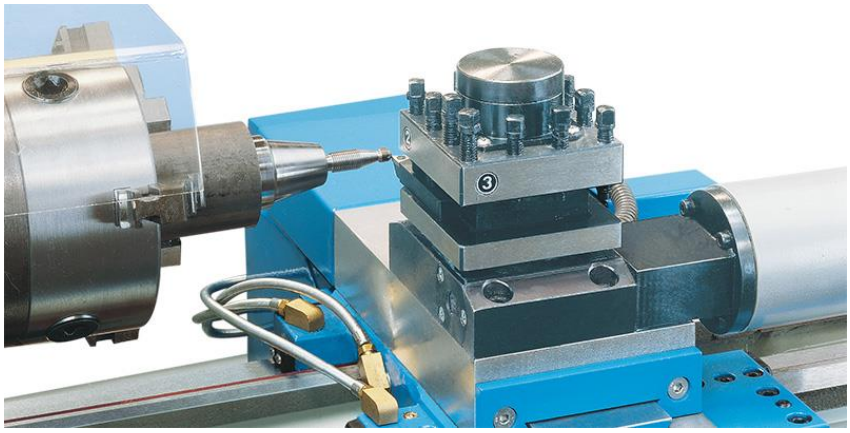
G es la dirección correspondiente a las funciones preparatorias. Se utilizan para informar al control de las características de las funciones de mecanizado. La función G va seguida de un número de dos cifras que permite programar hasta 100 funciones preparatorias diferentes.

M corresponde a nominación de funciones auxiliares, tales como parada de la máquina, activación de la refrigeración, etc.

En un principio se usan la tarjetas perforadas como forma de control de las maquinas herramienta, pero el modo de funcionamiento ha cambiado desde entonces tomando una dirección computarizada. El principio de funcionamiento sigue siendo el mismo; para mecanizar una pieza se usa un sistema de coordenadas que rigen el movimiento de la máquina herramienta. Mediante el software de programación ejecutado por computador, se controlan los movimientos de la máquina herramienta según los ejes de coordenadas asignadas.

En el caso de un torno, se deben controlar los movimientos de la máquina herramienta en las coordenadas X, para desplazamientos laterales del carro, y Z, para desplazamientos transversales de la torre. Para ambos desplazamientos se utilizan servomotores.

Figura 1. Torno de control numérico



Fuente: Torno de control numérico [En línea] [Fecha de consulta 16 de marzo de 2012] Disponible en: <http://www.knuth.de/produkt,25296,sprache,4.html>

En el caso de las fresadoras, se controla el movimiento vertical en la coordenada Z. Para esto se incorpora un servomotor en el mecanismo de desplazamiento ubicado en la mesa.

Figura 2. Fresadora de control numérico

Fuente: Fresadora de control numérico [En línea] [Fecha de consulta 16 de marzo de 2012] Disponible en: <http://www.gratis-tutoriales.com.ar/tornos-cnc/fresadora-cnc-roland-comercial>

1.1 CARACTERISTICAS DEL CONTROL NUMERICO

Con una máquina herramienta CNC, se obtiene un producto final de calidad superior, debido a su fácil control, alta precisión y flexibilidad en cuanto a modificaciones en el concepto de la pieza a fabricar; dando la posibilidad de elaborar piezas complejas de manera sencilla; inclusive, el proceso de corte puede ser simulado con el propósito de constatarlo. Ya que este tipo de máquinas son prácticamente automáticas, un solo operador que puede o no tener experiencia en el manejo de maquinaria CNC, está en capacidad de manipular algunas máquinas

al mismo tiempo, haciendo más segura su labor y menos fatigante. Debido a que la línea de producción se hace más uniforme, se facilita el control de calidad del producto, se reduce costos de inventario, traslado, fabricación; incluso se puede entregar el producto casi inmediatamente si así lo requiere la demanda; esto permite administrar el proceso de producción de manera asequible. [1]

1.2 EVOLUCIÓN DEL CONTROL NUMÉRICO

A continuación se describe la evolución del control numérico:

1725: En Inglaterra, se construyen máquinas de tejer controladas por tarjetas perforadas.

1863: M. Forneaux, el primer piano que se tocó de forma automática.

1870-1890: Desarrollo de plantillas y dispositivos.

1880: Variedad de herramientas para trabajar metales. Inicio del énfasis en la producción a escala.

1940: Aparecen los controles hidráulicos, neumáticos y electrónicos. Énfasis en maquinado automático.

1942: La Bendix Corporation tiene inconvenientes en fabricar una leva tridimensional, puesto que su perfil es imposible de realizar con máquinas accionadas manualmente. Una maquina accionada automáticamente pudo definir un gran número de puntos en su trayectoria.

1945: Se inicia la investigación y desarrollo del control numérico. Primera producción a escala con control numérico.

1947: Jhon Parsons crea un mando automático. Utiliza tarjetas perforadas en un lector para traducir las señales de mando. Primera fresadora de tres ejes en contorneado mandado por un control digital.

1953: El M.I.T. utiliza por primera vez la apelación “numerical control”.

1955: Aparecen la maquinas automatizadas en las plantas de producción de la fuerza aérea de Estados Unidos.

1956: Enfoque en el control numérico para trabajos más simples, tales como taladrado, mandrilado y punteado que no requieren de un movimiento continuo sino de un movimiento mucho más preciso. Esto revoluciona el proceso de fabricación.

1960: Primeras demostraciones de control adaptable. Este control permite la autorregulación de las condiciones de trabajo de la máquina.

1968: Primeros ensayos de control numérico directo.

1969-Actualidad: Nuevos sistemas de control numérico. Producción de gama más grande. Enfoque en diferentes tipos de materiales. Se comenzó a utilizar insumos computarizados. Se comenzó a utilizar documentos computarizados de control de gráficos. [2]

2. CODIGOS G

Las funciones preparatorias, también conocidas como G-Codes o Códigos G, son las más importantes en la programación CNC, ya que controlan el modo en que la máquina va a realizar un trazado, o el modo en que va a desplazarse sobre la superficie de la pieza que está trabajando.

Los posibles valores que acompañan a este comando, van de 00 a 99, y cada uno tiene una función determinada. Los más importantes, o al menos aquellos se observan a continuación. [3]

Tabla 1. Valores de comandos de códigos G.

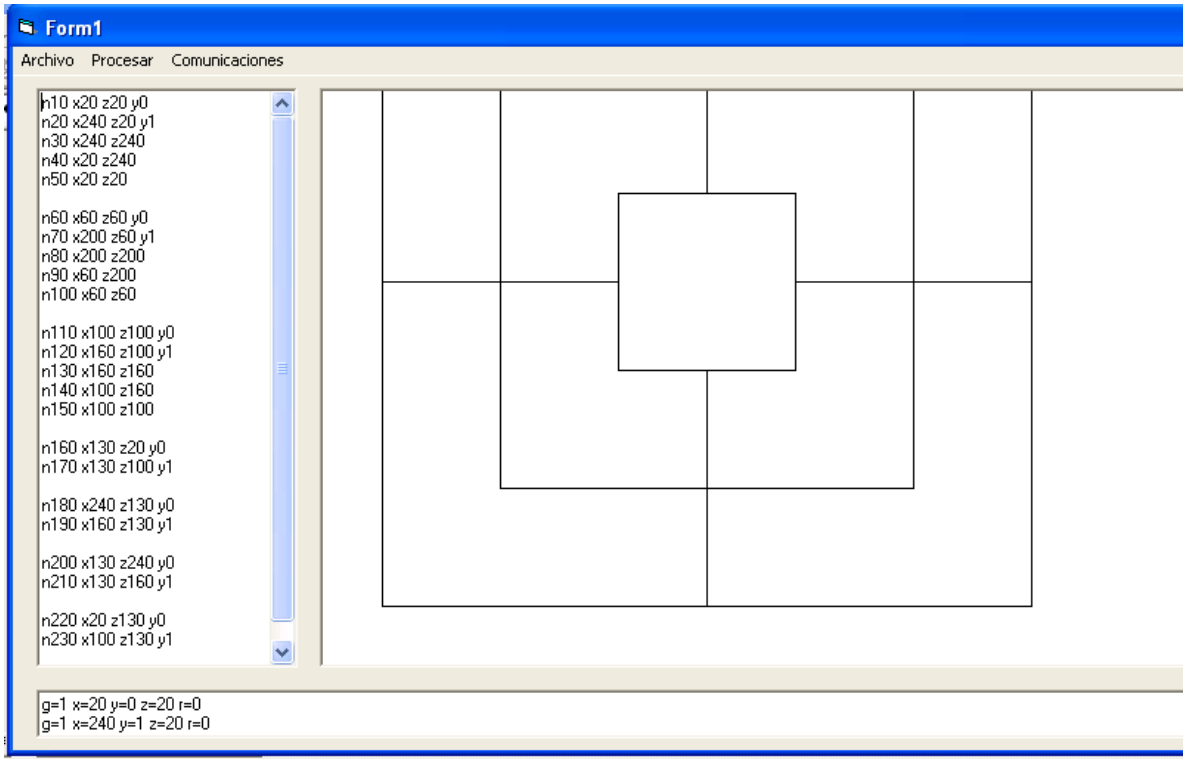
Comando	Descripción
G00	Interpolación Lineal Rápida.
G01	Interpolación lineal a la velocidad programada en el registro F.
G02	Movimiento Circular en el sentido horario Feedrate.
G03	Movimiento Circular en el sentido anti-horario Feedrate.
G04	Es una demora o una pausa con un tiempo específico.
G17	Selección del Plano X-Y
G18	Selección del Plano X-Z
G19	Selección del Plano Y-Z
G40	Compensación anulada, o al centro de la línea de desplazamiento.
G41	Compensación a la Izquierda de la línea de desplazamiento.
G42	Compensación a la Derecha de la línea de desplazamiento.

G70	Unidad de Datos expresados en Pulgadas.
G71	Unidad de Datos expresados en Milímetros.
G90	Desplazamiento en Modo Absoluto.
G91	Desplazamiento en Modo Incremental o Relativo.

Fuente: Tabla de valores de comandos de códigos G [En línea] [Fecha de consulta 16 de marzo de 2012] Disponible en: <http://r-luis.xbot.es/cnc/codes03.html>

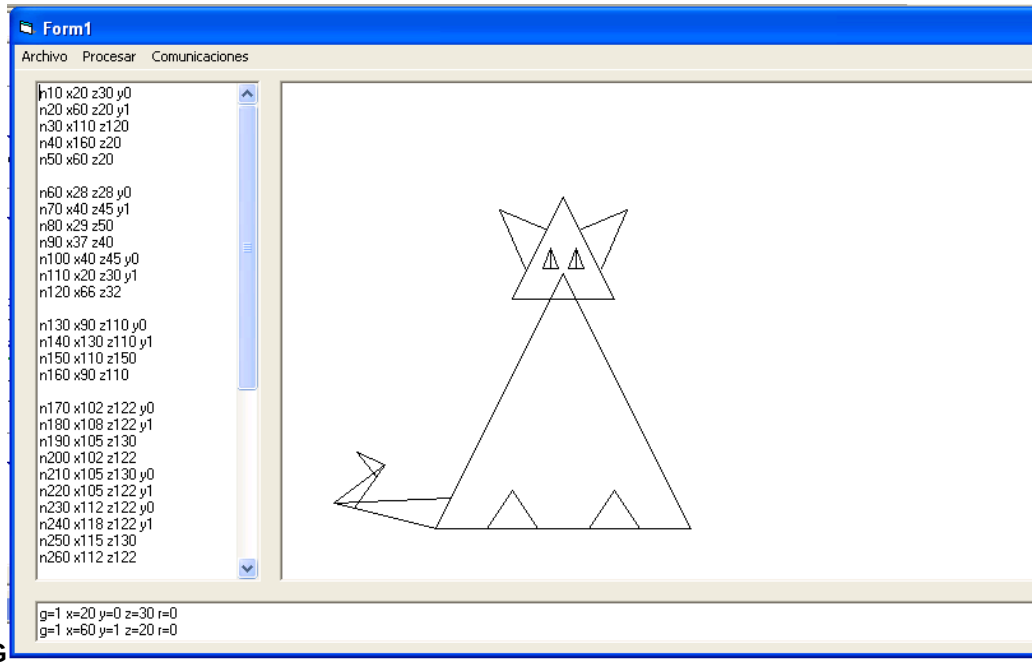
A continuación se muestran ejemplos de dibujos implementando códigos G:

Figura 3. Ejemplo 1 Códigos G



Fuente: Autores

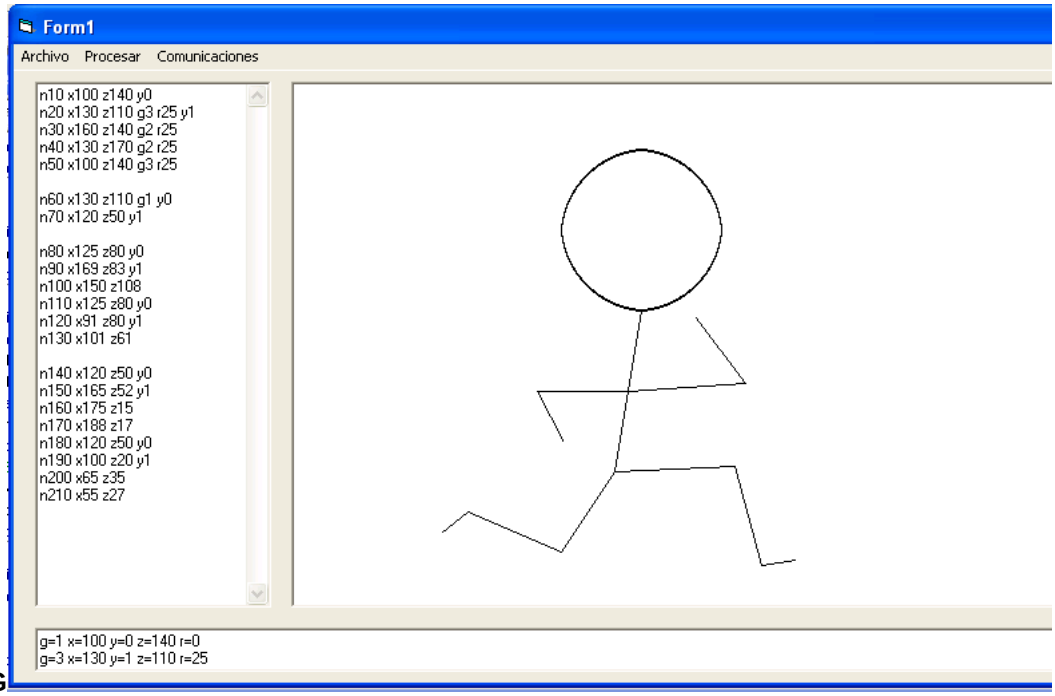
Figura 4. Ejemplo 2 Códigos



G

Fuente: Autores.

Figura 5. Ejemplo 3 Códigos



G

Fuente: Autores

3. MOTORES PASO A PASO

Los motores paso a paso son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos.

La característica principal de estos motores, es el hecho de poder moverlos un paso a la vez por cada pulso que se le aplique. Este paso puede variar desde 90° hasta pequeños movimientos de tan solo 1.8°, es decir, que se necesitarán 4 pasos en el primer caso (90°) y 200 para el segundo caso (1.8°), para completar un giro completo de 360°.

Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas están energizadas, el motor

estará enclavado en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas.

El motor paso a paso está constituido esencialmente por dos partes: a) Una fija llamada "estator", construida a base de cavidades en las que van depositadas las bobinas que excitadas convenientemente formarán los polos norte-sur de forma que se cree un campo magnético giratorio. b) Una móvil, llamada "rotor" construida mediante un imán permanente, con el mismo número de pares de polos, que el contenido en una sección de la bobina del estator; este conjunto va montado sobre un eje soportado por dos cojinetes que le permiten girar libremente. [4]

Figura 6. Motor paso a paso usado en la mesa de dibujo

Fuente: Autores

Para el movimiento en el eje X y en el eje Z de la mesa de dibujo se usaron 2 (dos) motores paso a paso de 24 (veinticuatro) voltios como el de la figura 3.

Para el control de los motores paso a paso se usa el PLC TWIDO de TELEMECANIQUE referencia TWDLCAE40DRF, por medio de este se controla la posición y velocidad de los motores en la mesa de dibujo de máquinas de control numérico.

Se usa un circuito de potencia que consiste en un arreglo de transistores TIP 31C en modo corte-saturación que permite aumentar la corriente de salida del PLC hacia los motores.

Las especificaciones de los motores son:

Corriente del motor:

$$I_m = 2A$$

Resistencia del motor:

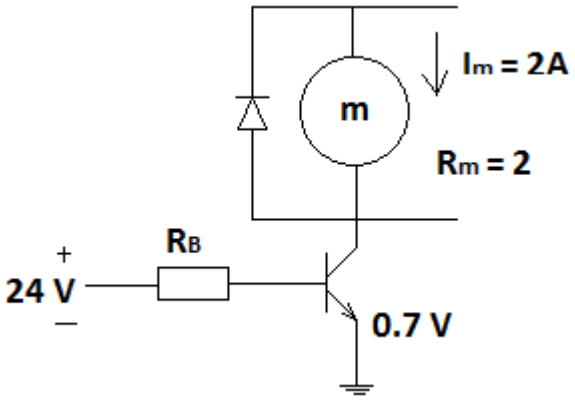
$$R_m = 2.06\ \Omega$$

Voltaje de alimentación del motor:

24V

El circuito de potencia usado es:

Figura 7. Circuito de potencia



Fuente: Autores

A partir de la Figura 4. Correspondiente a Circuito de potencia se realizan los cálculos correspondientes para hallar la resistencia de base del transistor denominada R_B

En donde:

$$I_c = 2A$$

$$I_B = \frac{I_c}{0.7\ \beta}$$

$$I_B = \frac{2}{0.7 * 40}$$

$$I_B = 71.4mA$$

Corriente de salida del transistor

$$I = 500mA$$

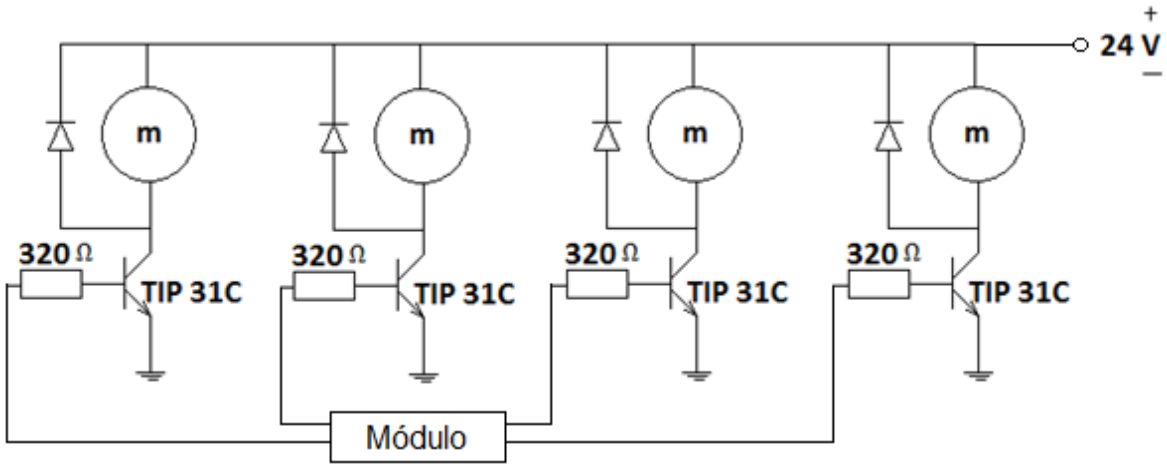
Entonces:

$$R_B = \frac{24 - 0.7}{71.4\ mA}$$

$$R_B = 326\Omega$$

El circuito de potencia completo para cada motor es:

Figura 8. Circuito de potencia motores paso a paso



Fuente: Autores.

4. MÓDULO DE SALIDAS DE TRANSISTOR DIGITALES

Se usa el módulo de salidas de transistor digitales referencia TM2DDO8TT para controlar los transistores de los motores paso a paso de la mesa de dibujo dándoles mayor velocidad, durabilidad mecánica y eléctrica.

4.1 DEFINICIÓN

El TM2DDO8TT es un módulo de salida de transistor de 8 canales. Este módulo está equipado con un bloque de terminales de tornillos de conexión extraíble que sirve para conectar salidas [7].

4.2 DESCRIPCIÓN GENERAL

Las características principales del módulo TM2DDO8TT son[7]:

- 8 canales de salida.
- 1 línea común para 8 canales.
- 0.5 A máximos por salida.
- 4 A máximos por línea común.
- 24 V CC tensión de salida.
- De 20.4 a 28.8 V CC es su rango de tensión de salida.
- 0.4 V CC máximo en caída de tensión
- 450 µs tiempo de encendido
- 450 µs tiempo de apagado
- Su protección de salida contra sobrecorriente y cortocircuito es de una limitación de I< 1,7 A, desconexión de las 8 salidas y reinicio automático cuando se reduce la temperatura.

5. MESA DE DIBUJO

5.1 ESTRUCTURA BASICA DEL SISTEMA

Figura 9. Estructura básica del sistema

Fuente: Autores

Como se observa en la Figura 9, la mesa de dibujo funciona por medio del siguiente proceso:

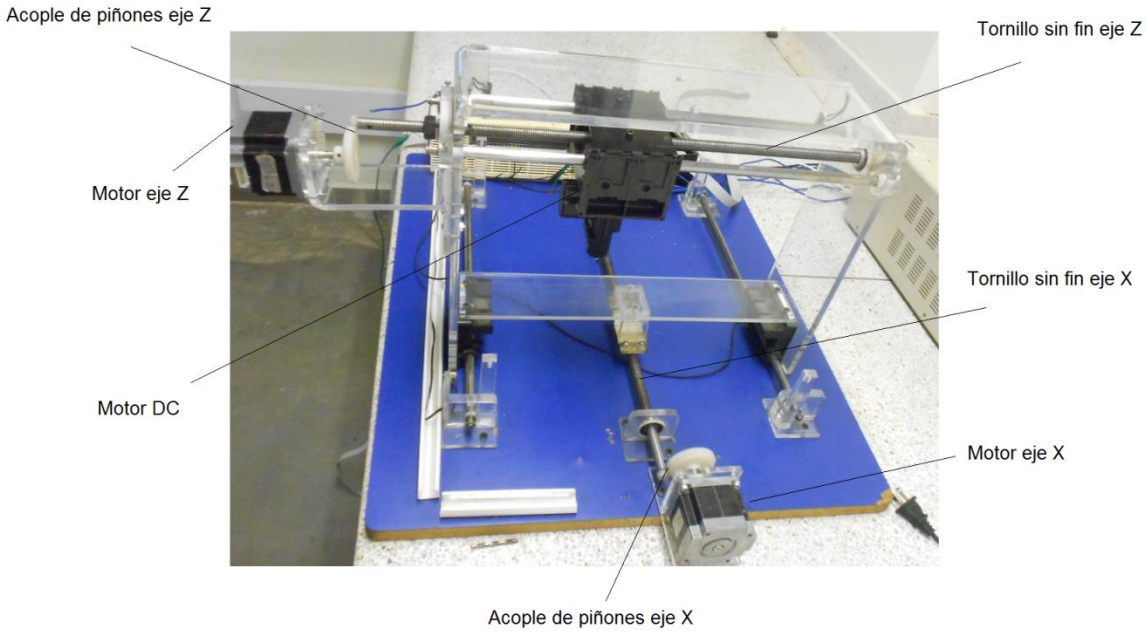
1. Se realiza el dibujo a plasmar en la mesa de dibujo, en la interfaz gráfica previamente desarrollada en Visual Basic.
2. Se establece comunicación entre el computador (interfaz gráfica) y el controlador lógico programable PLC TWIDO TELEMECANIQUE, esta comunicación es establecida mediante el protocolo Modbus TCP/IP.
3. El PLC TWIDO TELEMECANIQUE controla el circuito de potencia por medio del modulo de salidas transistorizadas TM2DDO8TT.
4. El circuito de potencia se encarga de proporcionar el movimiento y control de los motores que mueven la estructura, en este caso la mesa de dibujo, en los ejes X y Z.

5.2 DESCRIPCION GENERAL

Para el desarrollo de la parte mecánica de este trabajo de grado, se usa la mesa de dibujo existente en el laboratorio de máquinas eléctricas de la Universidad Pontificia Bolivariana, a esta maqueta se le realizan ciertas modificaciones entre las que se encuentran el cambio de motores de corriente continua a motores paso a paso, el control de la mesa de dibujo por PLC y las comunicaciones establecidas por medio de protocolo Modbus TCP/IP.

La mesa de dibujo consta básicamente de 2 (dos) motores paso a paso que controlan los ejes X y Z, un motor de corriente continua que controla el eje Y, 4 finales de carrera que se encargan de delimitar el área de dibujo y el área de dibujo como se observa en la siguiente figura:

Figura 10. Mesa de dibujo



Fuente: Autores

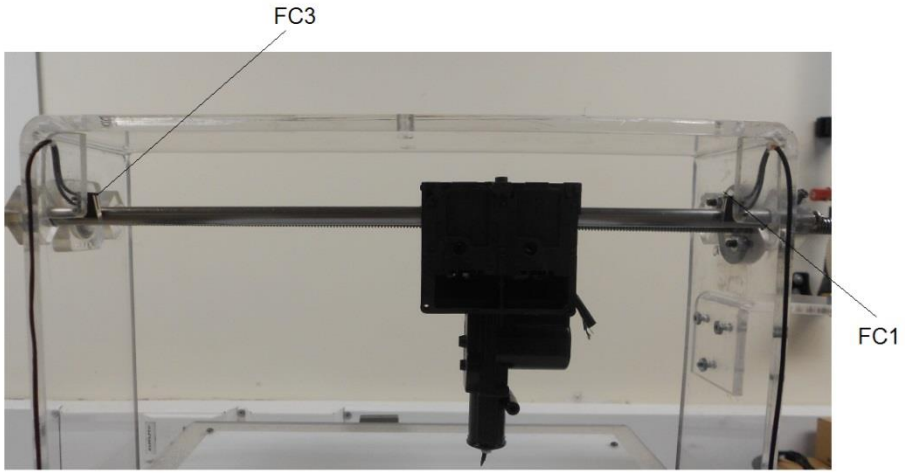
El motor en el eje X esta acoplado a un sistema de piñones con el fin de multiplicar la velocidad del paso dado a la hora de plasmar un dibujo, la multiplicación conseguida es 4 veces mayor a la normal, este acople de piñones se realiza por medio de un piñón pequeño 20 (veinte) dientes unido a un tornillo sin fin que mueve toda la estructura del eje Z, y otro piñón de 120 (ciento veinte) que está unido al motor paso a paso, al realizar el acople se garantiza que el tornillo sin fin va a mover la estructura 4 veces más rápido que si se uniera el motor directamente al tornillo sin fin como se había realizado inicialmente.

El motor del eje Z tiene el mismo arreglo de piñones mencionado anteriormente para el eje X con la diferencia que el tornillo sin fin en este eje está encargado de mover el carro en el que está ubicado el motor de corriente continua que controla el eje Y.

El motor de corriente continua encargado de controlar el eje Y está unido al lápiz de dibujo, este es el encargado de plasmar la figura sobre el papel en el área de dibujo, que proviene de la interfaz gráfica, este motor solo se energiza cuando el lápiz baja a dibujar lo que se controla dentro de la programación de la interfaz gráfica, de lo contrario el motor esta des energizado.

Los 4 (cuatro) finales de carrera se encargan de detener el funcionamiento de la mesa cuando los ejes X y Z llegan a su punto inicial 0.0 (cero punto cero) al igual que cuando alcanzan su punto máximo dentro del área de dibujo, este punto máximo se alcanza en 25 (veinticinco) centímetros en X y 25 (veinticinco) centímetros en Z. La ubicación de los finales de carrera es la siguiente:

Figura 11. Finales de carrera eje Z



Fuente: Autores

Figura 12. Finales de carrera eje X

Fuente: Autores

- El final de carrera llamado FC1 en la Figura 11 es el control del punto inicial o punto 0 (cero) del eje Z.
- El final de carrera llamado FC2 en la Figura 12 es el encargado del control del punto máximo en el eje X.
- El final de carrera llamado FC3 en la Figura 11 es el control del punto inicial o punto 0 (cero) del eje Z.
- El final de carrera llamado FC4 en la Figura 12 es el encargado del control del punto máximo en el eje X.

6. CONTROLADOR LOGICO PROGRAMABLE PLC

6.1 DEFINICIÓN DEL PLC

Un controlador lógico programable PLC es un dispositivo electrónico ideal para la automatización de líneas de producción, en general, ya que por medio de su configuración y sincronización se pueden realizar los controles de diferentes procesos y maquinarias a nivel industrial.

El PLC elegido para la realización de este proyecto fue el PLC TWIDO TELEMECANIQUE modelo compacto referencia TWDLCAE40DRF.

6.2 DESCRIPCIÓN GENERAL DEL PLC

Las características principales de este PLC TWIDO TELEMECANIQUE referencia TWDLCAE40DRF son:

- Modelo compacto de 24 entradas digitales
- 14 salidas de relé
- 2 salidas de transistor
- 2 potenciómetros analógicos
- 1 puerto de serie integrado
- 1 slot para un puerto de serie adicional
- RTC integrado
- Compartimiento de batería para batería externa reemplazable por el usuario
- Admite hasta 7 módulos de ampliación de E/S.
- Admite hasta dos módulos de interface del bus AS-Interface V2
- Admite un módulo master de interface del bus de campo CANopen

- Admite un cartucho de memoria opcional (de 32 ó 64 KB)
- Admite un módulo de monitor de operación opcional
- Puerto RJ45 de interface Ethernet integrado

6.3 LENGUAJES DE PROGRAMACIÓN DEL PLC

La programación del PLC TWIDO TELEMECANIQUE referencia TWDLCAE40DRF se realiza por medio del software de programación TwidoSuite que es un entorno de desarrollo gráfico para crear, configurar y mantener aplicaciones para autómatas programables Twido[8][6].

Para crear programas de control Twido se pueden utilizar los siguientes lenguajes de programación [5][6]:

- Lenguaje de lista de instrucciones: Consiste en una serie de expresiones lógicas escritas como una secuencia de instrucciones booleanas.

- Diagramas Ladder Logic: Es una forma gráfica de mostrar una expresión lógica.

- Lenguaje Grafcet: Está compuesto por una sucesión de pasos y transiciones. Twido admite las instrucciones de lista Grafcet, pero no Grafcet gráfico.

6.4 PROTOCOLO DE COMUNICACIÓN DEL PLC

Los controladores Twido disponen de un puerto serie, o de un segundo puerto opcional, que se utiliza para servicios en tiempo real o de administración de sistemas. Los servicios en tiempo real proporcionan funciones de distribución de datos para intercambiar datos con dispositivos de E/S, así como funciones de

administración para comunicarse con dispositivos externos. Los servicios de administración de sistemas controlan y configuran el controlador por medio de TwidoSuite. Cada puerto serie se utiliza para cualquiera de estos servicios, pero sólo el puerto serie 1 es válido para comunicarse con TwidoSuite.

Para poder utilizar estos servicios, existen tres protocolos disponibles en cada controlador [5][6]:

- Conexión remota: consiste en bus maestro/esclavo de alta velocidad, diseñado para transferir una pequeña cantidad de datos entre el controlador master hasta un máximo de siete controladores esclavos remotos.

- ASCII: permite establecer comunicaciones entre el controlador y un dispositivo simple como, por ejemplo, una impresora

- Modbus: El protocolo Modbus es un protocolo maestro/esclavo que permite a un único maestro solicitar respuestas de los esclavos o realizar acciones dependiendo de las peticiones. El maestro puede dirigirse a los esclavos particulares o iniciar una difusión de mensajes para todos los esclavos

El protocolo de comunicación utilizado en este proyecto de programación e implementación de una mesa de dibujo para máquinas de control numérico fue el protocolo Modbus TCP/IP debido a que la referencia TWDLCAE40DRF proporciona un puerto de comunicación RJ45 Ethernet integrado.

6.5 PROGRAMA DESARROLLADO PARA LA MESA DE DIBUJO DE MÁQUINAS DE CONTROL NUMÉRICO

Usando el software TwidoSuite como entorno de programación se desarrolla el siguiente programa para el control de la mesa de dibujo.

Se escoge el lenguaje de programación LADDER del PLC TWIDO TELEMECANIQUE, para la mesa de dibujo de máquinas de control numérico. El lenguaje LADDER es un lenguaje gráfico que tiene como principal ventaja que sus símbolos básicos esta normalizados según el estándar IEC 3113(International Electrotechnical Commission) y son usados por todos los fabricantes.

A continuación se explicara el programa escalón por escalón, para mejor entendimiento se especifican en la siguiente tabla todas las variables usadas:

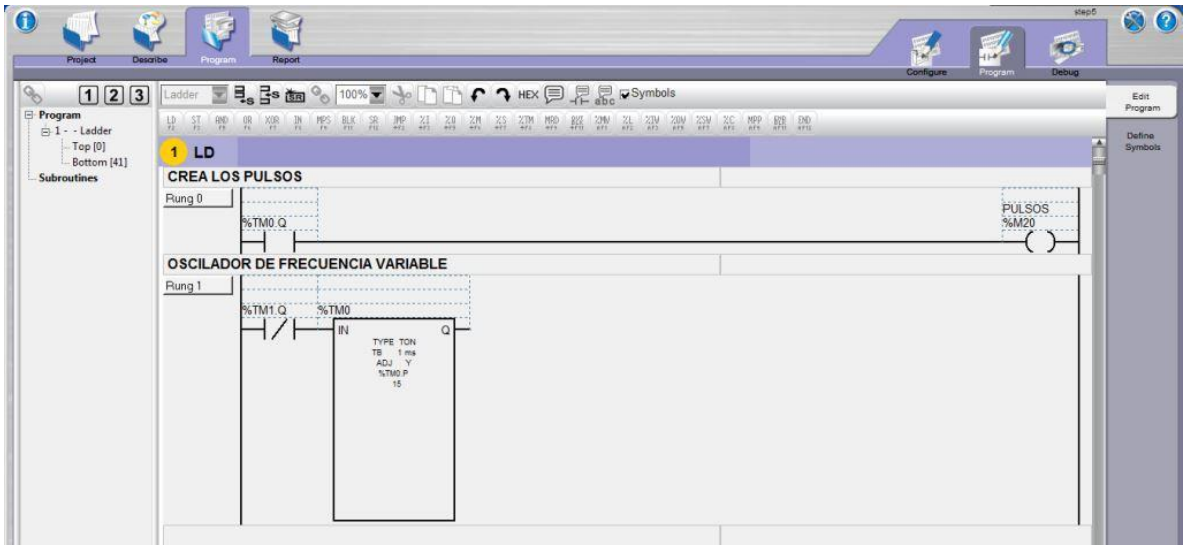
Tabla 2. Especificación de variables

Variable PLC	Definición de la Variable	Acción de variable
%I0.12	PX	Final de carrera eje X
%I0.13	PZ	Final de carrera eje Z
%M20	PULSOS	Indica velocidad de movimiento de los motores
%M21	MAYORX	Indica cuando el valor de X es mayor
%M22	MENORX	Indica cuando el valor de X es menor
%M23	MAYORZ	Indica cuando el valor de Z es mayor
%M24	MENORZ	Indica cuando el valor de Z es menor
%M25	HOMEX	Indica inicio en X
%M26	HOMEZ	Indica inicio en Z
%M27	HOMMING	Habilita HOMEX y HOMEZ

%M29	P_ALCX	Punto alcanzado en X
%M30	P_ALCZ	Punto alcanzado en Z
%M31	IGUAL	Indica igualdad entre valor actual en X y valor final en X
%M32	DIFERENTE	Indica operaciones del parámetro de la recta
%M33	P_ALCANZ	Indica que X y Z han llegado al punto deseado
%MF40	ZI	Valor inicial en Z
%MF42	ZF	Valor final en Z
%MF44	XI	Valor inicial en X
%MF46	XF	Valor final en X
%MF48	A_B	Auxiliar de los cálculos de la recta
%MF50	M	Valor de la pendiente m
%MF52	B	Valor del desplazamiento de la recta
%MF54	DX	Diferencial de Z
%MF56	DZ	Diferencial de X
%MF58	Z	Valor actual en Z
%MF60	X	Valor actual en X
%MW0	POSICION_X	Dato que indica el movimiento del dibujo
%MW1	POSICION_Z	Dato que indica movimiento del dibujo
%MW31	P_ENDZ	Punto final en Z
%MW32	TRAYEX	Trayectoria en X
%MW33	AUXPX	Mueve motor en X
%MW35	P_ENDX	Punto final en X
%MW36	TRAYEZ	Trayectoria en Z
%MW37	AUXPZ	Mueve motor en Z

Fuente: Autores

Figura 13. Oscilador de frecuencia variable

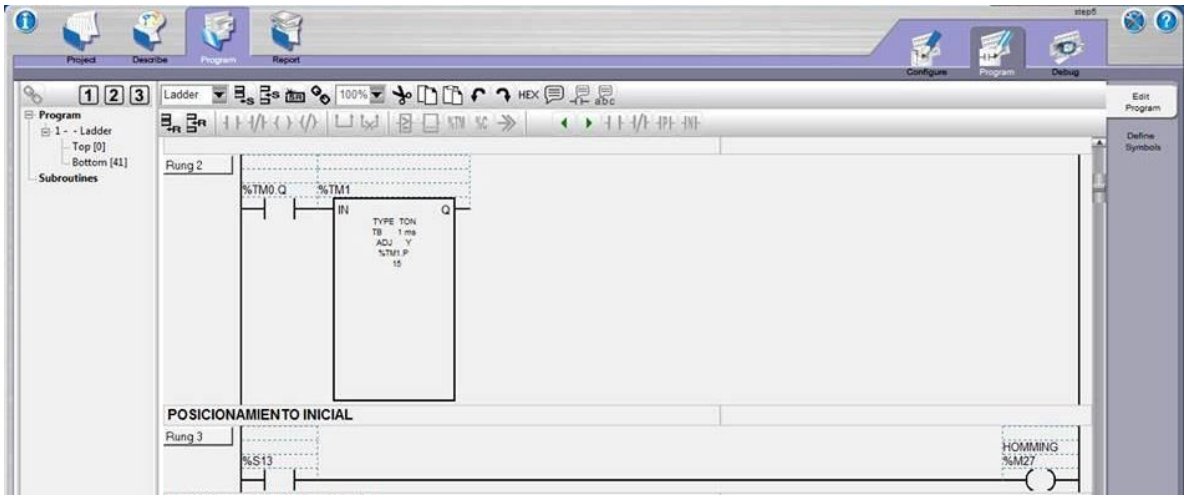


Fuente: Autores

Se inicia con 2 (dos) temporizadores, los cuales generan los pulsos de control de movimiento de los dos motores paso a paso usados en la mesa de dibujo, uno de los motores para el eje X y el otro motor para el eje Z. En la figura 4 se observa el primero de estos temporizadores en el escalón Rung 1 mientras que el segundo se observa en la figura 5 en el escalón Rung 2.

Los dos temporizadores denominados %TM0 y %TM1 manejan la bobina PULSOS que corresponde a la variable %M20 ubicada en el escalón Rung 0 la cual controla la velocidad de movimiento de los motores.

Figura 14. Posicionamiento inicial

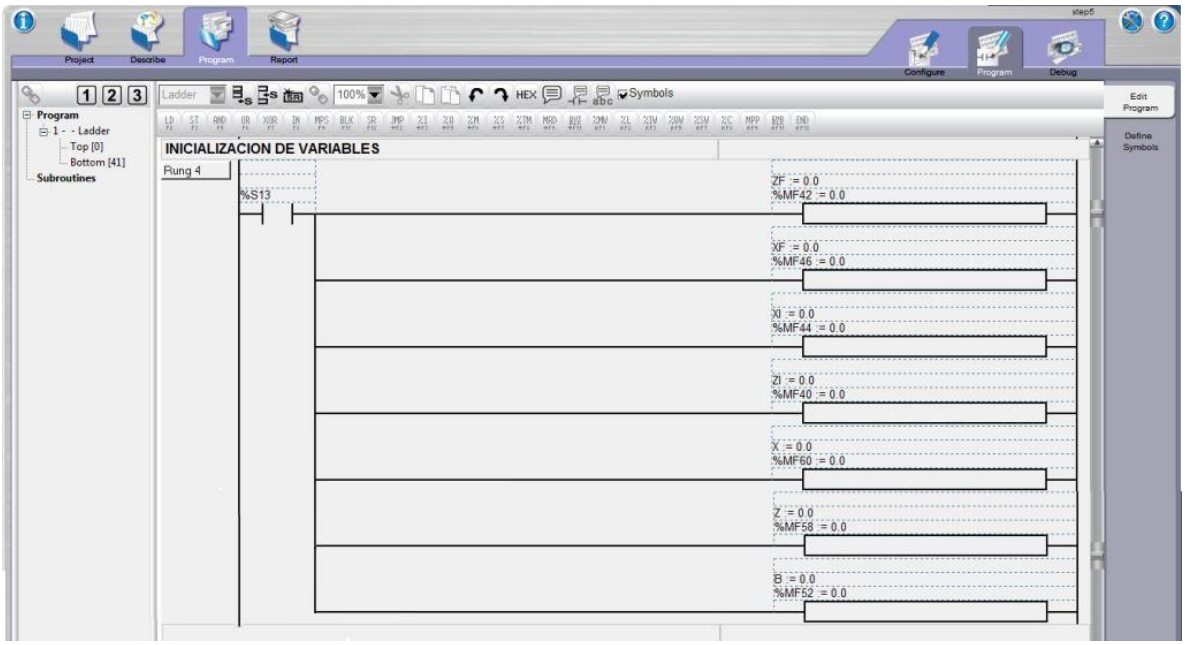


Fuente: Autores

Rung 3: El bit de sistema %S13 cumple la función de hacer el primer ciclo de ejecución. Este bit normalmente se encuentra en estado 0 (cero) y cambia a estado 1 (uno) durante el primer ciclo completo cuando el programa se encuentra en ejecución (RUN).

Este bit habilita la bobina HOMMING que es usada para darle a la mesa un posicionamiento inicial, es decir, que apenas se ejecuta el programa este siempre va a volver al punto de inicio 0,0 (cero coma cero).

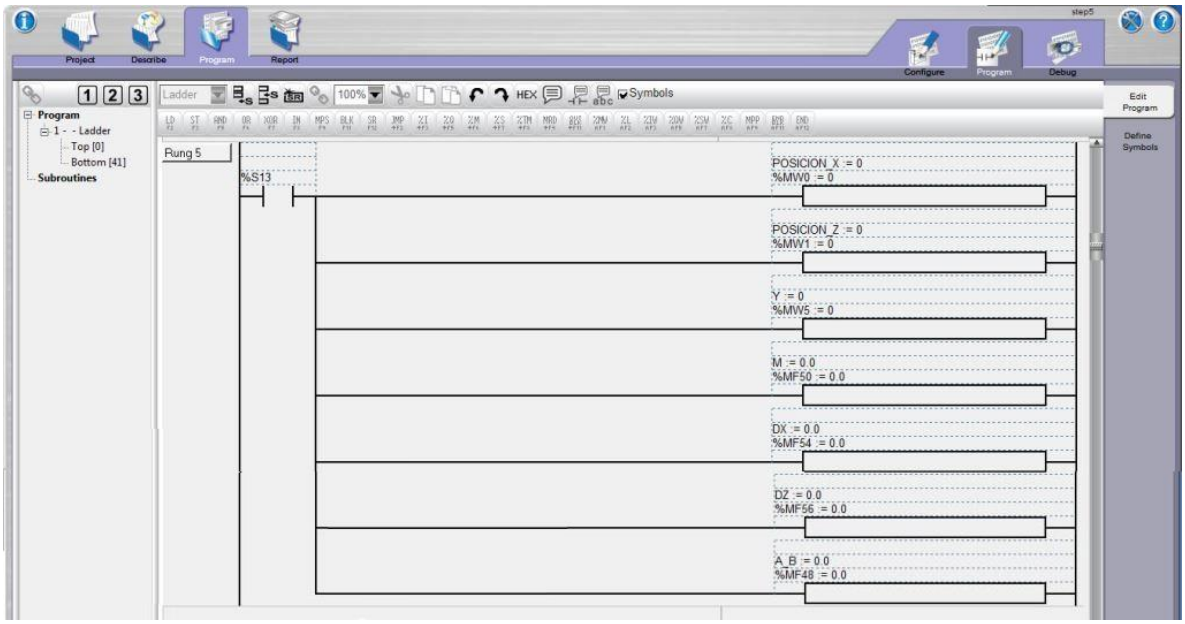
Figura 15. Inicialización de variables



Fuente: Autores

Rung 4: Nuevamente se usa el bit de inicialización %S13 para llevar las variables a 0 (cero). Esta inicialización de variables se realiza, para mejor funcionamiento del programa debido a que el PLC TWIDO de TELEMECANIQUE, en ocasiones puede guardar datos en sus variables; lo cual genera error a la hora de ejecutar el programa de manera repetitiva.

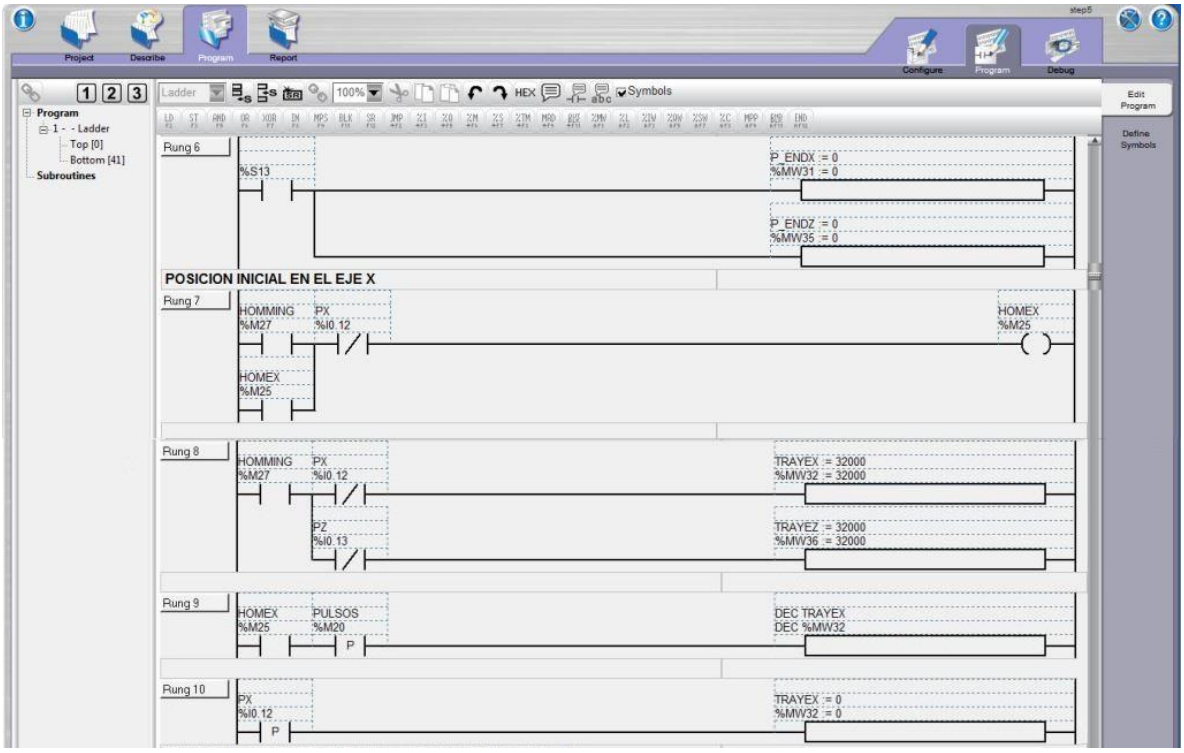
Figura 16. Inicialización de variables



Fuente: Autores

Rung 5: Se alcanza el máximo de líneas dentro del escalón Rung 4 así que se realiza más inicialización de variables en este escalón Rung 5.

Figura 17. Posición inicial eje X



Fuente: Autores

Rung 6: En este escalón se registran las últimas inicializaciones del sistema.

Rung 7: Se observa un contacto normalmente abierto que es controlado por la bobina HOMMING visto en la Figura 5, Rung 3. Esto habilita la bobina HOMEX, cuya función es llevar el motor del eje X a la posición inicial 0 (cero). Este posicionamiento inicial se realiza para garantizar, que independiente del punto en el que se encuentre el lápiz de dibujo cuando se inicie el programa, este vaya al inicio a esperar instrucciones del programa para realizar el dibujo.

Se tiene un final de carrera (PX) en la posición 0 (cero) para detenerlo cuando llegue al inicio.

Rung 8: Nuevamente se usa el contacto normalmente abierto HOMMING, esta vez para habilitar dos bloques de operaciones en donde se iguala TRAYEX y TRAYEZ a 32000 (treinta y dos mil), siendo TRAYEX la trayectoria recorrida por el motor en el eje X y TRAYEZ la trayectoria recorrida por el motor en el eje Z.

Se usa el número 32000 (treinta y dos mil), ya que este garantiza que, sin importar la posición en la que se encuentre el lápiz de dibujo, va a llegar a su posición inicial 0,0 (cero coma cero).

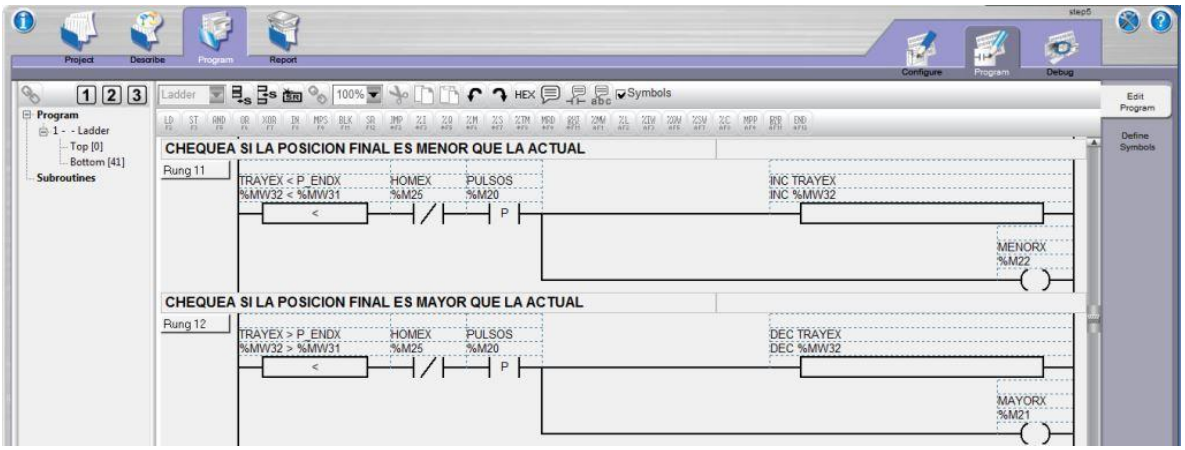
Este número de 32000 es el máximo alcanzado por la mesa de dibujo.

Se usan dos finales de carrera denominados PX y PZ, como contactos normalmente cerrados, para garantizar que el valor 32000 no cargue de nuevo una vez llegado al inicio.

Rung 9: El contacto normalmente abierto llamado HOMEX se encuentra habilitando un bloque de operaciones de nombre DEC TRAYEX, cuya función es decrementar el valor de la trayectoria recorrida por el motor del eje X desde 32000 (treinta y dos mil) hasta la posición inicial, ya sea que llegue a 0 (cero) o se active final de carrera PX en el inicio del eje X.

Rung 10: La función del final de carrera PX ubicado en la posición 0 (cero) del eje X es darle el valor de 0 (cero) a la trayectoria en X es decir carga un valor de 0 (cero) en la variable TRAYEX sin importar el valor que haya decrementado hasta el momento.

Figura 18. Comparación de posiciones



Fuente: Autores

Rung 11: El bloque de comparación denominado:

$$TRAYEX < P\_ENDX$$

Determina si la posición final es menor que la actual, lo que activa el bloque de operaciones denominado INC TRAYEX, que incrementa el trayecto recorrido por el motor del eje X. A su vez se activa la bobina MENORX.

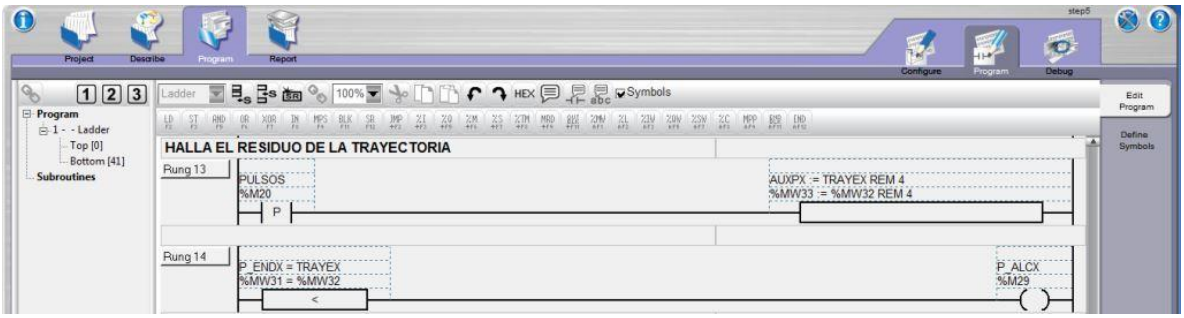
Rung 12: Este bloque de comparación hace lo mismo que el anterior pero al contrario, es decir, comprueba si la posición final es mayor que la actual, lo que activa el sentido contrario del motor con el bloque de operaciones llamado DEC TRAYEX, que decrementa el trayecto recorrido por el mismo motor. A su vez se activa la bobina MAYORX.

Estos bloques de comparaciones se usan para comprobar el sentido del giro del motor, es decir, si el siguiente movimiento en el eje X es positivo o negativo respecto al punto anterior.

Los movimientos, ya sean de incremento o decremento, son solo posibles si ya se han cumplido las condiciones iniciales, es decir, si HOMEX ya ha llegado al punto 0 (cero) del eje X. A su vez ambos movimientos están condicionados por la

velocidad dada a la bobina PULSOS que controla la velocidad de los pasos del motor.

Figura 19. Encuentra el residuo de la trayectoria



Fuente: Autores

Rung 13: La operación aritmética REM, es usada en el bloque de operaciones de la siguiente manera:

$$AUXPX := TRAYEX\ REM\ 4$$

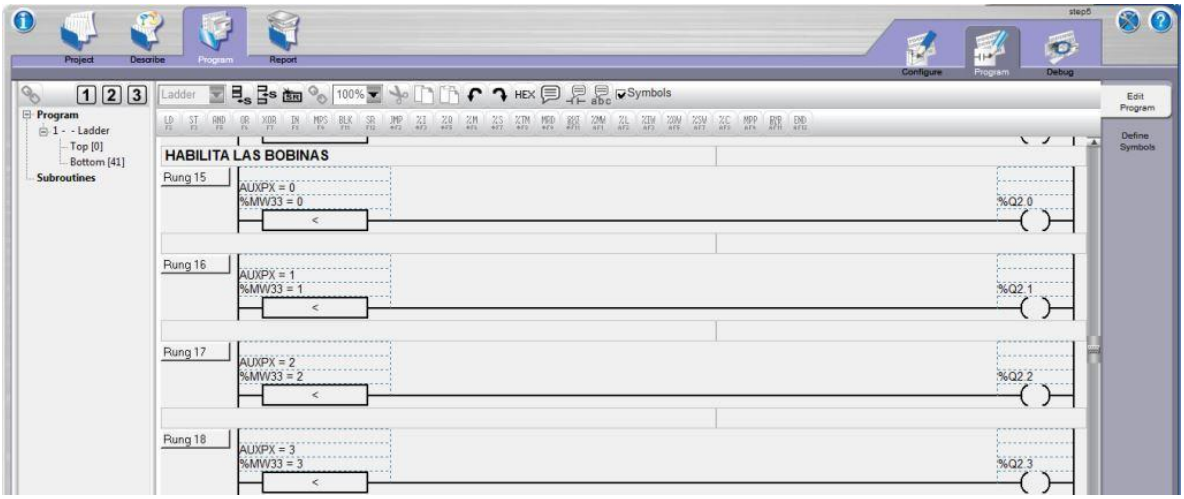
Esto indica que el residuo de la división de TRAYEX entre 4 (cuatro) queda guardada en la variable auxiliar AUXPX. Esta operación está condicionada por la velocidad de la bobina PULSOS.

Rung 14: Se usa un bloque de comparación así:

$$P\_ENDX = TRAYEX$$

Esta operación determina si la posición final ya es igual a la actual, lo que activa la bobina llamada P_ALCX, es decir, que ha alcanzado el punto en X.

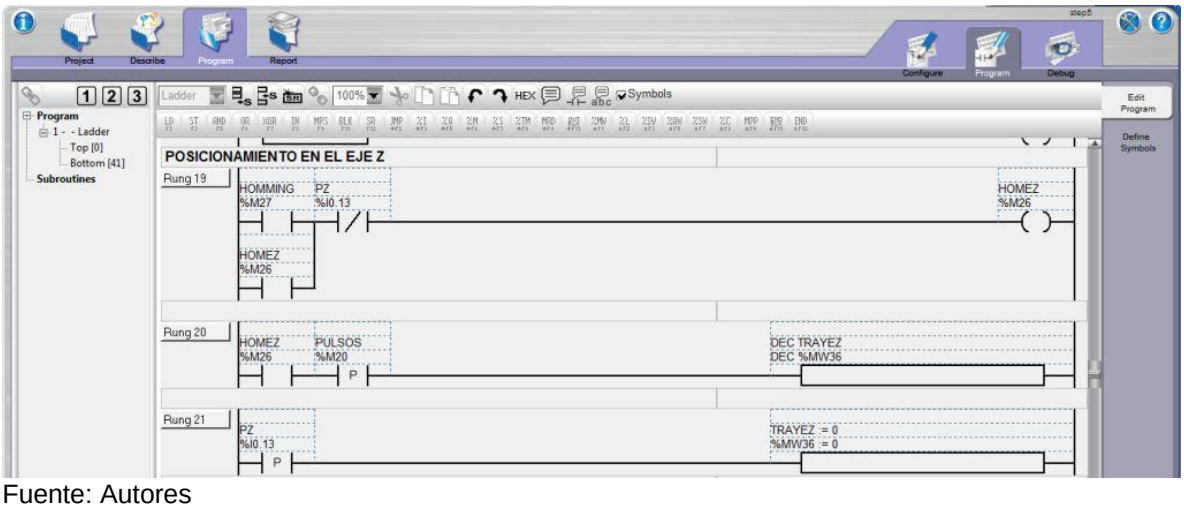
Figura 20. Mueve el motor en X



Fuente: Autores

Rung 15, Rung 16, Rung 17, Rung 18: Dependiendo del valor resultante en AUXPX de la operación aritmética anterior, la que se encuentra ubicada en Figura 10, Rung 13, se le asigna el valor correspondiente a 4 de las salidas del módulo de expansión TM2DDO8TT. Estas salidas son las que activan los pines de conexión del motor asignado garantizando que dé el paso a la velocidad dada por la operación. El orden de conexión de los pines del motor debe ser asignado previamente.

Figura 21. Posicionamiento en el eje Z



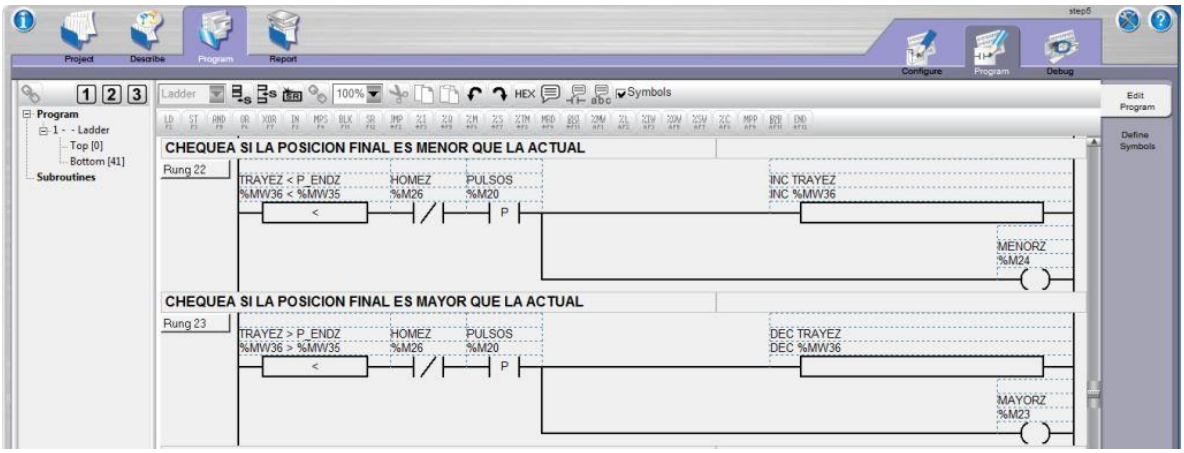
Fuente: Autores

Rung 19: Se observa que el contacto normalmente abierto, que es controlado por la bobina HOMMING, está habilitando la bobina HOMEZ, cuya función es mover el motor en el eje Z y llevarlo hasta el punto inicial 0 (cero). Una vez más esto se hace para garantizar las condiciones iniciales sin importar el lugar en el que se encuentre el lápiz de dibujo al momento de ejecutar el programa.

Rung 20: El contacto normalmente abierto controlado por la bobina HOMEZ habilita el bloque de operaciones DEC TRAYEZ, cuya función es decrementar el valor guardado en TRAYEZ hasta llevarlo al punto inicial desde 32000 (treinta y dos mil) hasta que se encuentre con el final de carrera PZ ubicado en el punto inicial 0 (cero) del eje Z. Esta operación está condicionada por la velocidad con la que se alimenta la bobina PULSOS.

Rung 21: El final de carrera PZ se ubica al inicio del eje Z y al activarse le da un valor a TRAYEZ de 0 (cero) inmediatamente, sin importar el valor al que haya decrementado hasta ese momento, garantizando así las condiciones iniciales del programa a ejecutar.

Figura 22. Detección de posiciones mayor y menor



Fuente: Autores.

Rung 22: En un bloque de comparación, se comprueba si la posición final es menor que la posición actual, mediante la operación siguiente:

$$TRAYEZ < P\_ENDZ$$

Esta operación significa que si TRAYEZ en menor que el punto al que se necesita llegar P_ENDZ se debe incrementar su valor hasta que sean iguales. Entonces, si se cumple esta operación se activa el bloque de operación INC TRAYEZ. Se activa también la bobina MENORZ.

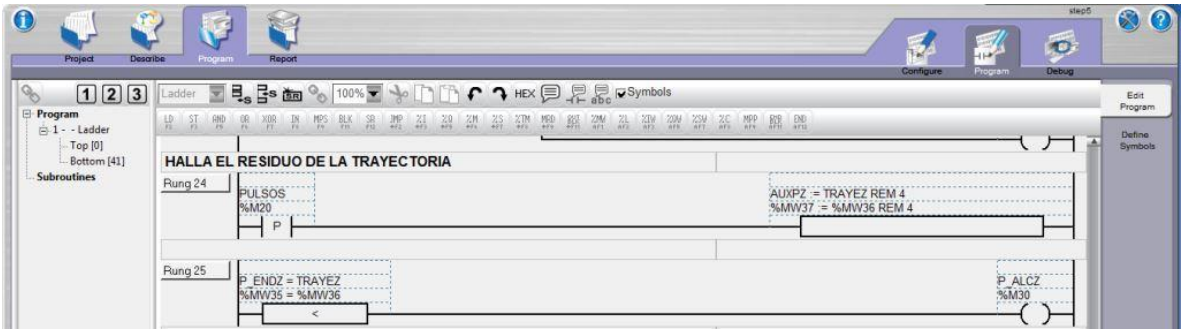
Rung 23: En otro bloque de comparación se hace la comprobación contraria, es decir, se comprueba si la posición final es mayor que la posición actual mediante la operación siguiente:

$$TRAYEZ > P\_ENDZ$$

Esto significa que si el valor de la trayectoria TRAYEZ en el momento actual es mayor que el punto al que se desea mover el motor en el eje Z, se debe decrementar el valor de TRAYEZ hasta que alcance el valor que se desea, es decir, que se igual a P_ENDZ. También se activa la bobina MAYORZ.

Ambos escalones, tanto el de incremento como el de decremento, dependen de si ya se han cumplido las condiciones iniciales del programa, es decir, si HOMEZ ya ha llegado al inicio 0 (cero) en el eje Z al ejecutar el programa. También están condicionados por la velocidad aplicada a la bobina PULSOS.

Figura 23. Halla residuo de la trayectoria



Fuente: Autores.

Rung 24: Como se vio en la Figura 10, Rung 13, se aplica la operación aritmética REM, que da el valor del residuo de una división entre dos operandos. Esta operación se aplica a la división de TRAYEZ entre 4 (cuatro) y el valor resultante se guarda en la variable auxiliar AUXPZ. Entonces el bloque de operación queda de la siguiente manera:

$$AUXPZ = TRAYEZ\ REM\ 4$$

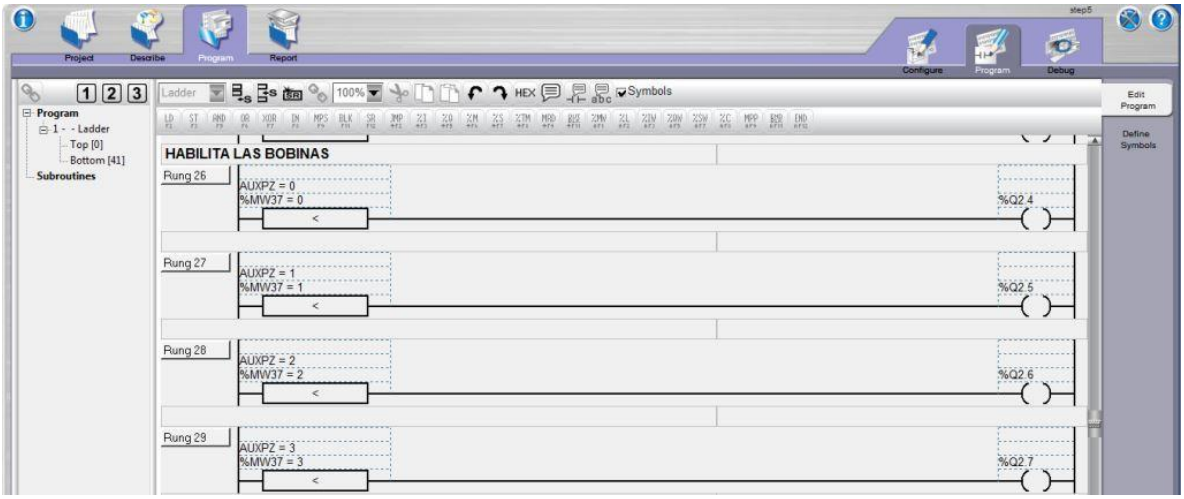
Esta operación está condicionada por el valor de velocidad aplicada a la bobina PULSOS.

Rung 25: En un bloque de comparación así:

$$P\_ENDZ = TRAYEZ$$

Se comprueba si el valor de la trayectoria en el eje Z, es decir la variable TRAYEZ, ya ha llegado al punto de que se necesita mostrado en la variable P_ENDZ. Si esta comparación se cumple se activa la bobina P_ALCZ.

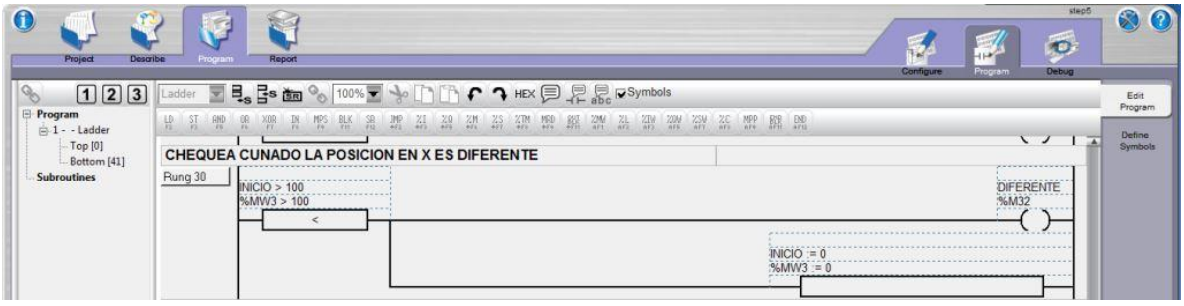
Figura 24. Habilita bobinas motor en Z



Fuente: Autores

Rung 26, Rung 27, Rung 28, Rung 29: Dependiendo del valor guardado en la variable auxiliar AUXPZ en la operación de la Figura 14, Rung 24, se le asigna el valor correspondiente a las 4 salidas restantes del módulo de expansión del PLC TM2DDO8TT. Estas salidas controlan los pines del motor en el eje Z y garantizan el movimiento sobre todo el eje, ya sea positivo o negativo. Los pasos del motor están controlados por la velocidad dada a la bobina PULSOS.

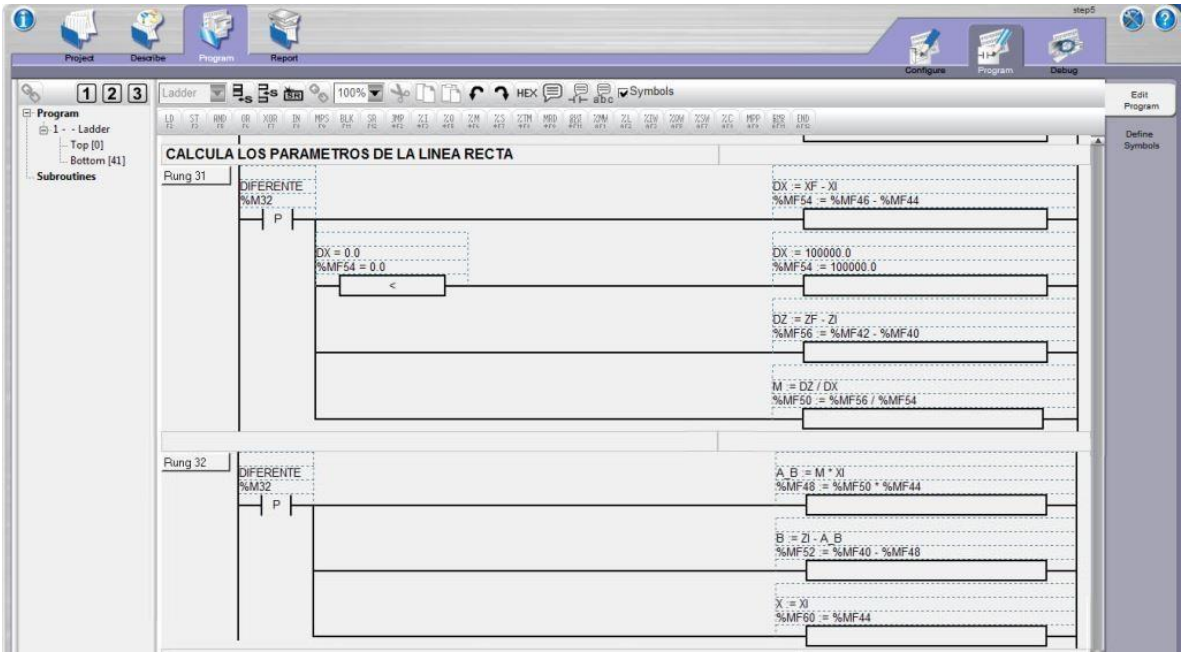
Figura 25. Chequeo de posición en X



Fuente: Autores

Rung 30: Se coloca un bloque de comparación que comprueba si INICIO es mayor a 100. Esta comparación se hace a manera de START (o inicio) para garantizar que en cada movimiento o trazado del programa se ejecute y no se quede en error. Después de activar la bobina llamada DIFERENTE, el mismo se reinicia a 0 (cero), esperando así que en el siguiente movimiento se vuelva a cumplir la condición INICIO > 100 para asegurar el movimiento.

Figura 26. Cálculo de línea recta



Fuente: Autores

Rung 31: Un contacto normalmente abierto controlado por la bobina DIFERENTE envía un pulso que activa las operaciones para encontrar los parámetros de la recta que se quiere trazar en los bloques de operaciones.

Se realizan los cálculos de línea recta que deben ser seguidos por el PLC TWIDO TELEMECANIQUE usando las fórmulas generales de la línea recta las cuales corresponden a:

$$y = mx + b$$

$$d_X = X_f - X_i$$

$$d_Y = Y_f - Y_i$$

$$m = \frac{d_Y}{d_X}$$

$$aux = mX_i$$

$$b = Y_i - aux$$

Aplicando estas fórmulas generales en los bloques de operaciones del programa quedan de la siguiente manera:

$$D_X = X_F - X_I$$

$$D_Z = Z_F - Z_I$$

$$M = \frac{D_Z}{D_X}$$

En este caso se usó una línea de seguridad con un bloque de comparación que pregunta si el DX es igual a 0 (cero). Si esta condición es cierta al hacer la operación de la pendiente de la recta, que es M = DZ / DX, el programa generaría un error debido a la indeterminación de la división por 0 (cero). Para solucionar este inconveniente se planteó que si se cumple la condición, DX=0, automáticamente el programa cambia este valor de 0 (cero) por un número muy alto, en este caso 100000 (cien mil), de esta manera la pendiente M tomaría un

valor bastante pequeño, cercano a 0 (cero), y no afectaría la recta que se quiere plantear.

Rung 32: Un contacto normalmente abierto controlado por la bobina de nombre DIFERENTE controla los bloques de operación para hallar los datos restantes de la línea que se quiere trazar. De esta manera quedan los siguientes datos:

$$A\_B = M * X_I$$

$$B = Z_I - A\_B$$

$$X = X_I$$

Figura 27. Genera pulso cuando la posición es alcanzada

Fuente: Autores

Rung 33: Dos bloques de comparación son usados para comprobar si los puntos, tanto en el eje X como en el eje Z, han llegado a sus destinos. La primera comparación pregunta si TRAYEZ ya ha llegado al punto que el programa le ha solicitado, ya sea incrementando o decrementando, de la siguiente manera:

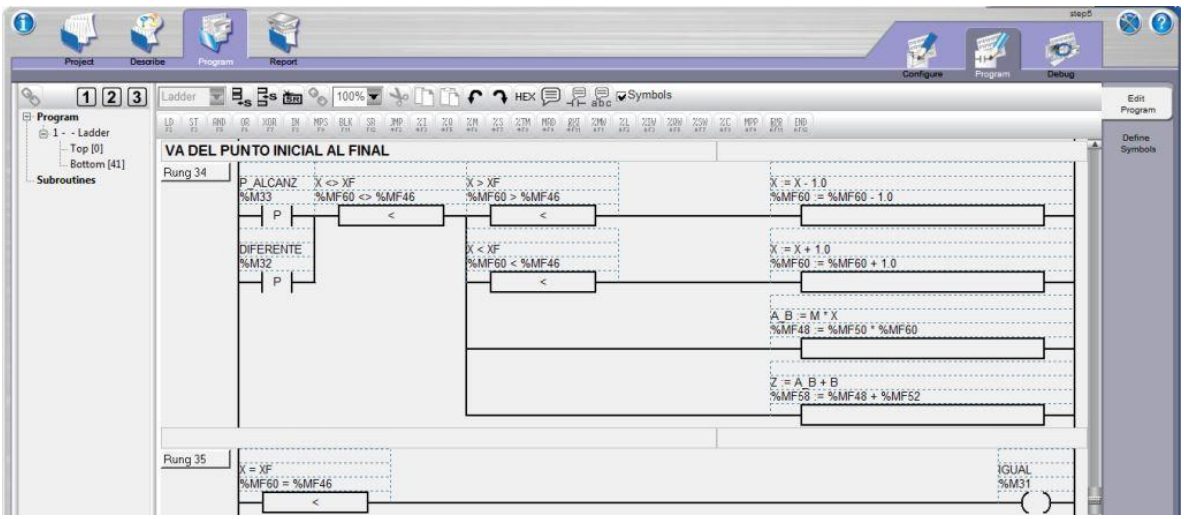
$$TRAYEZ = P\_ENDZ$$

La segunda comparación hace la misma pregunta pero en el eje X; pregunta si TRAYEX ya ha llegado al punto solicitado por el programa, incrementando o decrementando, de la siguiente forma:

$$TRAYEX = P\_ENDX$$

Si estas dos condiciones se cumplen se activa la bobina llamada P_ALCANZ que representa el punto en donde se ha alcanzado el valor pedido en ambos ejes de la mesa.

Figura 28. Comparación valores en X



Fuente: Autores

Rung 34: El contacto normalmente abierto de nombre P_ALCANZ, que corresponde a que el punto anterior ya ha llegado a su destino programado como se vio en la Figura 18, Rung 33, envía un pulso para hacer el movimiento de los motores junto con el contacto normalmente abierto de nombre DIFERENTE que se activa con la condición de INICIO > 100 como se vio en la Figura 16, Rung 30.

Cuando se cumple cualquiera de estos 2 (dos) pulsos, se coloca un bloque de comparación que comprueba que el valor de X actual sea diferente al valor del X final, XF, de la siguiente manera:

$$X <> X_F$$

Si esta condición no se cumple significa que X actual y el X final, XF, son iguales por lo tanto el punto anterior ya llego a su destino y está esperando nuevas instrucciones. Si esta condición se cumple significa que hay un nuevo movimiento requerido por el programa.

Se colocan dos nuevos bloques de comparación preguntando si el nuevo valor del X final, XF, es menor o mayor, respectivamente, que el valor de X actual. La comparación de hace de la siguiente manera:

$$X > X_F$$

Y

$$X < X_F$$

Si la primera de estas comparaciones se cumple significa que el nuevo valor del X final, XF, es menor que el valor de X actual, por lo que deberá decrementar el valor de X actual en el bloque de operaciones de la siguiente manera:

$$X = X - 1.0$$

Si por el contrario la condición que se cumple es la segunda significa que el nuevo valor del X final, XF, es mayor que el valor de X actual, por lo que se debe

incrementar el valor de X actual en el bloque de operaciones de la siguiente manera:

$$X = X + 1.0$$

Una vez hechas estas operaciones, independientemente si incremento o decremento, se pasa a hacer los cálculos del valor de Z en los siguientes bloques de operaciones de la siguiente manera:

$$A\_B = M * X$$

$$Z = A\_B * B$$

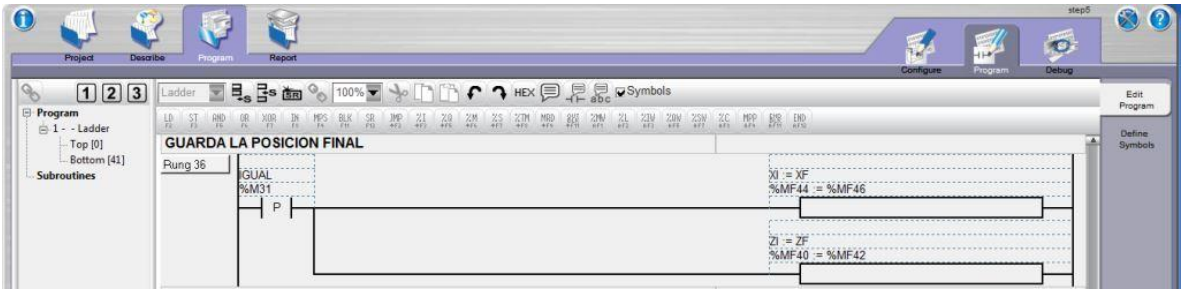
Teniendo los valores de X y Z se puede proceder al trazo solicitado por el programa.

Rung 35: Un bloque de comparación comprueba que el valor actual de X y el valor del x final, XF, sean iguales. Esto significa que el valor de X actual ha llegado a su destino final. Esta comparación se hace de la siguiente manera:

$$X = X_F$$

Si se cumple esta condición se activa la bobina llamada IGUAL.

Figura 29. Guarda la posición final



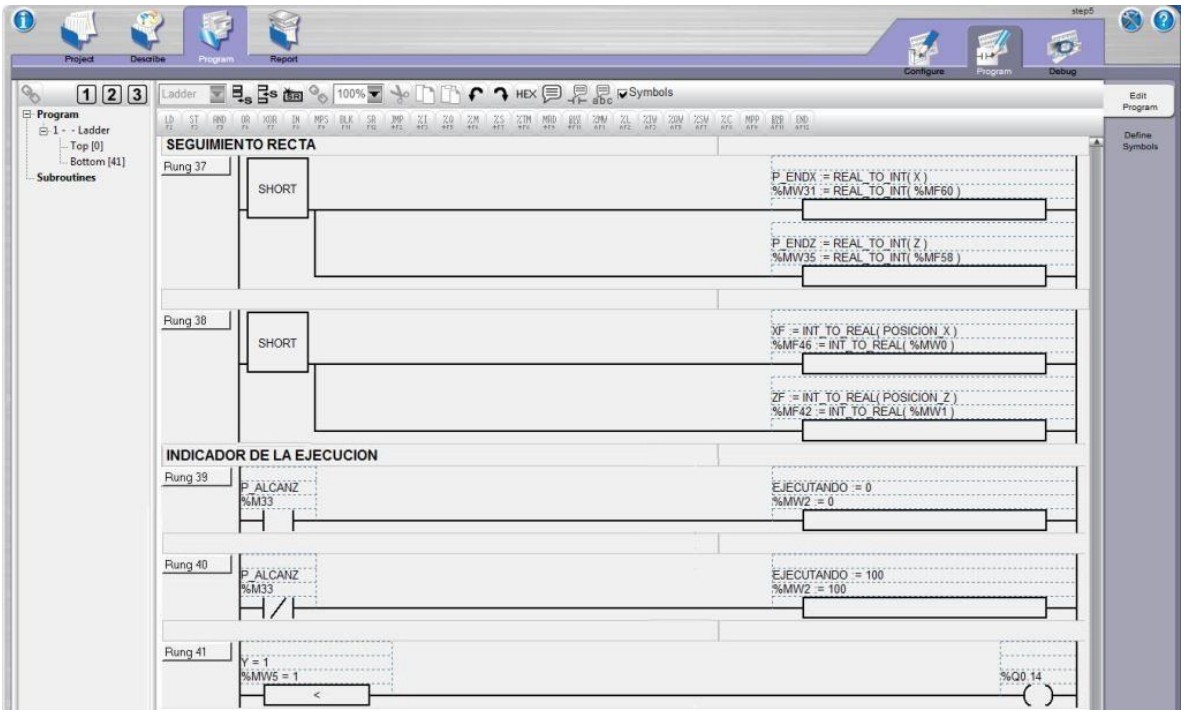
Fuente: Autores.

Rung 36: En la Figura 19, Rung 35 se vio como se activaba la bobina IGUAL, así una vez activado se tiene un contacto normalmente abierto que envía un pulso e iguala los valores finales con los valores iniciales de la siguiente manera:

$$X_I = X_F$$

$$Z_I = Z_F$$

Figura 30. Conversión de variables



Fuente: Autores

Rung 37: Antes que nada se debe entender la diferencia entre objetos de palabra y objetos flotantes. Los objetos de palabra, como su nombre lo indica, son aquellos que son enviados en forma de palabra de 16 bits, que pueden ser almacenados en la memoria interna y son números enteros. Los objetos flotantes son, también como su nombre lo indica, aquellos que permiten flotantes. Un flotante es un argumento matemático que posee un decimal en su expresión [4][5].

Ahora bien las operaciones entre objetos de palabra y objetos flotantes no se pueden realizar directamente, para ello es necesario que se convierta uno o más operandos de manera que todos los operandos sean del mismo tipo de objeto.

Para realizar operaciones enteras con números flotantes es necesario convertir el número flotante a entero y eso se hace de la siguiente forma:

$$P\_ENDX = REAL\_TO\_INT(X)$$

Siendo el valor de X un número flotante y el valor de la variable P_ENDX un número entero. De esta forma ya es posible hacer operaciones con otros valores enteros. Así mismo:

$$P\_ENDZ = REAL\_TO\_INT(Z)$$

Siendo el valor de Z un número flotante y el valor de la variable P_ENDZ un número entero.

Rung 38: Una vez más es necesario hacer la conversión pero esta vez en sentido contrario. Entonces se tiene:

$$X_F = INTO\_TO\_REAL(POSICION\_X)$$

Siendo el valor de POSICION_X un numero entero y el valor de XF un numero flotante. Así mismo:

$$Z_F = INTO\_TO\_REAL(POSICION\_Z)$$

Siendo el valor de POSICION_Z un numero entero y el valor de ZF un numero flotante.

Rung 39, Rung 40: Estos escalones se usan como indicadores visuales de la ejecución, de manera que el contacto normalmente abierto llamado P_ALCANZ hablita el bloque de operación y hace la variable EJECUTANDO igual a 0 (cero).

En el caso contrario, cuando no se ha llegado al punto deseado, es decir, cuando el contacto normalmente cerrado llamado P_ALCANZ este activo, se coloca un valor de 100 (cien) en la variable EJECUTANDO.

La función de la variable EJECUTANDO es el monitoreo del movimiento de los motores cuando se les asigna los valores de X y Z que se desea dibujar.

Rung 41: Por último, se tiene el motor en Y. Este motor controla el movimiento de subida y bajada del lápiz de dibujo. Se coloca un bloque de comparación donde se pregunta si el valor de Y esta en 1 (uno). Si se cumple esta condición se habilita un pin de salida del PLC llamado %Q0.14, que energiza el motor en Y generando el trazo deseado por el programa.

7. VISUAL BASIC

7.1 DEFINICION

Entorno de programación compilador y creador de aplicaciones graficas de fácil entendimiento e interacción con el usuario.

Visual Basic 6.0 es el entorno de programación usado para crear el software de control para la mesa de dibujo de máquinas de control numérico.

Visual Basic 6.0 no puede ser desarrollada en Windows Vista, Windows 7 o Windows server 2008 por lo cual, se instala una máquina virtual WNDOWS XP MODE para la ejecución del software desarrollado y su interfaz gráfica implementada en la mesa de dibujo para máquinas de control numérico.

7.2 DESARROLLO DE INTERFAZ GRÁFICA Y PROGRAMACIÓN

La interfaz gráfica desarrollada en Visual Basic tiene como finalidad diseñar, crear, observar, editar, escribir y dibujar los diferentes esquemas que requiera el usuario en la mesa posicionadora con una simple lectura del desarrollo de los códigos G ingresados. Los pasos a seguir son los siguientes:

- Escribir los códigos G para la figura deseada.
- Dibujar el esquema cuando se haya finalizado.
- Editar los códigos G en caso de haber errores en el diseño.
- Dibujar el esquema definitivo.
- Guardar el esquema.
- Establecer comunicación con el PLC para envió de datos.

La interfaz gráfica está conformada por las siguientes partes desarrolladas en Visual Basic:

Figura 31. Interfaz gráfica

Fuente: Autores

- Menú: Es el lugar donde se pueden escoger todas las acciones a realizar en la interfaz. Se compone de Archivo, Procesar y Comunicaciones.
- Área de Código: Es el espacio en donde se escriben los códigos G para dibujar el esquema.
- Área de Dibujo: Es el lugar donde se muestra la figura hecha con los códigos G en el área de código.
- Indicadores: Muestra diferentes datos como la dirección TCP/IP del PLC a conectar, el número de líneas hecho en el área de código, el estado de la conexión en todo momento y la línea de código G que se encuentra ejecutando en ese mismo instante.
- Barra de Estados: Muestra el valor de los ejes X, Z y Y, así como los valores de G y R en cada línea de código G hecha en el área de código.

El menú se compone principalmente de 3 (tres) opciones así:

- Archivo: Es la parte encargada de la gestión de los archivos como guardar, cargar, nuevo y salir.
- Procesar: Es la parte encargada de la interpretación de los códigos G de los archivos como dibujar o limpiar.
- Comunicaciones: Es la parte encargada de entablar la comunicación entre el Visual Basic y el PLC con instrucciones como conectar, desconectar, leer, escribir, verificar y cancelar.

La descripción de cada una de las partes del menú al igual que su programación de describen a continuación:

7.2.1 Archivo

El menú archivo, que se encarga de gestionar los archivos creados por la interfaz. Se compone de los siguientes submenús:

7.2.1.1 Guardar

Se encarga de salvar en un archivo plano los códigos G escritos en el área de código. Este submenú aporta las facilidades que brinda Visual Basic para guardar los archivos en los diferentes dispositivos de almacenamiento que se encuentren conectados en ese momento.

El submenú GUARDAR se ejecuta de la siguiente manera:

- Se utiliza el objeto COMMON DIALOG de Visual Basic que crea una ventana capaz de permitir al usuario navegar entre carpeta para escoger el destino del archivo.
- Se configura la ventana creada por el COMMON DIALOG para que solo se muestren los archivos de texto plano.
- Se muestra la interfaz para realizar la búsqueda.
- Se abre el archivo escogido.
- Se escribe el código realizado en el archivo.
- Se cierra el archivo.

El código utilizado para este submenú es el siguiente:

Figura 32. Código submenú GUARDAR

```
Private Sub guardar_Click()
Dim VarTexto As String
Contenido = Text1.Text
CD1.Filter = "Ficheros de Texto |*.txt"
CD1.ShowSave
Open CD1.FileName For Output As #1
Print #1, Contenido
Close #1
End Sub
```

Fuente: Autores

7.2.1.2 Cargar

Permite cargar un archivo plano de códigos G, que haya sido previamente guardado en la interfaz gráfica o en algún programa editor de texto plano, con la finalidad de editarlo o ejecutarlo cuando se desee. Este submenú aprovecha las facilidades de Visual Basic para recuperar los archivos planos.

El submenú CARGAR se ejecuta de la siguiente manera:

- Se prepara la interfaz gráfica para que solo se puedan leer los archivos de texto plano.
- Se muestra la ventana donde se pueden manipular y gestionar los archivos por el usuario.
- Se abre el archivo seleccionado.
- Se lee todos los datos y códigos de una sola vez.
- Se cierra el archivo de texto.
- Se imprimen los códigos G del archivo de texto plano en el área de código.
- Se limpia el área de dibujo.

El código utilizado para este submenú es el siguiente:

Figura 33. Código submenú CARGAR

```
Private Sub cargar_Click()
Dim VarTexto As String
CD1.Filter = "Ficheros de Texto |*.txt"
CD1.ShowOpen
Open CD1.FileName For Input As #1
Contenido = Input(LOF(1), #1)
Close #1
'TBTexto = VarTexto
Text1.Text = Contenido
Picture1.Picture = Nothing
End Sub
```

Fuente: Autores.

7.2.1.3 Nuevo

El submenú NUEVO se encarga de limpiar y borrar todas las pantallas, es decir, las áreas de código y dibujo, así como todos los gráficos y textos escritos en la interfaz. Este submenú tiene como finalidad eliminar todo los datos anteriores para así empezar un nuevo proyecto desde cero.

El submenú NUEVO se ejecuta de la siguiente manera:

- Se imprime una cadena de texto vacía o nula sobrescribiendo la anterior en el área de código.
- Se grafica un objeto vacío o nulo sobrescribiendo el anterior en el área de dibujo.

El código utilizado para este submenú es el siguiente:

Figura 34. Código submenú NUEVO

```
Private Sub nuevo_Click()
Text1.Text = ""
Picture1.Picture = Nothing
End Sub
```

Fuente: Autores.

7.2.1.4 Salir

El submenú SALIR permite abandonar la interfaz gráfica y salir del entorno de Visual Basic.

El submenú SALIR se ejecuta de la siguiente manera:

- Cerrar y abandonar el entorno de desarrollo de Visual Basic.

El código utilizado para este submenú es el siguiente:

Figura 35. Código submenú SALIR

```
Private Sub salir_Click()
End
End Sub
```

Fuente: Autores.

7.2.2 Procesar

El menú procesar es el encargado de interpretar los códigos G escritos en el área de código, procesarlos por medio de una matriz donde se indiquen las coordenadas de los puntos solicitados en el esquema, así como sus formas de interpolación. Está compuesto por los siguientes submenús:

El menú PROCESAR se ejecuta de la siguiente manera:

- Establecer los códigos G que por omisión se van a trabajar en el desarrollo y procesamiento. Interpolación lineal G=1 y punta del lápiz alzada.
- Se leen los códigos G desde el área de código y se agrega un espacio en blanco en la última línea para una correcta interpretación del código.
- Se pasa todo a letras minúsculas.
- Se reemplazan los códigos especiales, como comas, en espacios en blanco para que no interfieran.
- En una cadena de igual longitud del código, se va leyendo uno por uno los caracteres.

    - Se obtiene cada uno de los caracteres por separado.
    - A partir de cada carácter, se crea una subcadena de texto.
    - Cuando se encuentre un carácter en blanco (espacio) se hace el análisis alfanumérico de los caracteres.

    Con la subcadena total se pasa a los siguientes pasos:

- Quitar los espacios en blanco en caso de haberlos.
- Obtener por separado la letra y el número de cada comando. La letra siempre está al inicio del comando y es el código G que se va a utilizar y el número es el restante que representa la coordenada.
- Una vez separado el número de coordenada de la letra en la subcadena, se convierte a flotante o a entero según se necesite.

  Con la letra de la subcadena, es decir, el código G, se obtiene la respectiva matriz:

  - Si el código G es una “n”, se crea la siguiente posición en la matriz. Se llena la interpolación lineal y la forma de escribir de la misma manera de la última modificación.
  - Si el código G es una “x”, se almacena el valor correspondiente en la matriz de coordenadas X.
  - Si el código G es una “z”, se almacena el valor correspondiente en la matriz de coordenadas Z.
  - Si el código G es una “y”, se almacena el valor correspondiente en la matriz de coordenadas Y.
  - Si el código G es una “g”, se cambia la forma de interpolación según la última modificación.
  - Si el código G es una “r”, se almacena el valor correspondiente en la matriz de coordenadas R.

El código desarrollado para el menú PROCESAR es el siguiente:

Figura 36. Código submenú PROCESAR

```
Private Sub procesar_Click()
Dim comandos, subcom As String
Dim i As Integer
j = 0
g = 1
y = 0
comandos = Text1.Text & " "
comandos = LCase(comandos)
comandos = Replace(comandos, vbNewLine, " ", , , vbTextCompare)
comandos = Replace(comandos, ",", " ", , , vbTextCompare)
subcom = ""
For i = 1 To Len(comandos)
    car = Mid$(comandos, i, 1)
    If car <> " " Then
        subcom = subcom & car
        Else
        subcom = Trim(subcom)
        letra = Left$(subcom, 1)
        If Len(subcom) > 0 Then
            numero = Right$(subcom, Len(subcom) - 1)
        End If
        cumint = Val(numero)
        Select Case letra
        Case "n": j = j + 1
                  valg(j) = g
                  valy(j) = y
                  valr(j) = 0
        Case "x": valx(j) = cumint
        Case "z": valz(j) = cumint
        Case "y": valy(j) = cumint
                  y = cumint
        Case "g": g = cumint
                  valg(j) = g
        Case "r": valr(j) = cumint
        End Select
        subcom = ""
        End If
```

Fuente: Autores

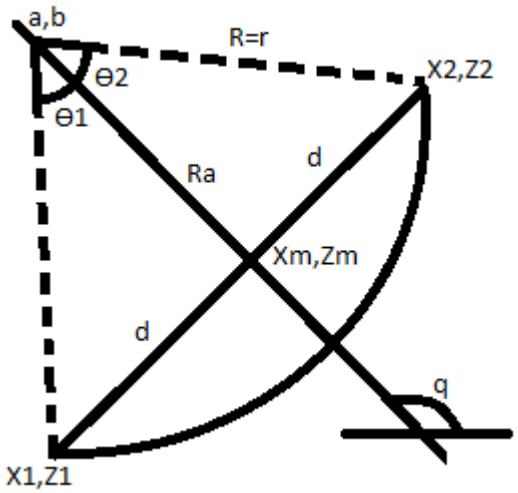
7.2.2.1 Dibujo

El submenú DIBUJO permite recrear los códigos G del área de código en el área de dibujo.

Este submenú se ejecuta de la siguiente manera:

- Se debe tener la matriz donde se tienen las coordenadas de los puntos a unir por el dibujo y su forma de interpolación.
- Las áreas de dibujo y de la mesa no son las mismas por lo que se deberá escoger la escala adecuada para el dibujo.
- En caso de que la forma de interpolación sea lineal, se deberá trazar una línea recta entre el punto actual y el siguiente punto.
- En caso de que la forma de interpolación sea circular y cóncava, se hace lo siguiente:

Figura 37. Interpolación circular.



Fuente: Autores.

Se quiere trazar un línea curva que comprende desde el punto (X1,Z1) hasta el punto (X2,Z2). El primer paso para el cálculo es trazar una recta entre estos dos puntos y encontrar un punto medio del trazo. Este cálculo se hace de la siguiente manera:

$$X_m = \frac{X_1 + X_2}{2}$$

$$Z_m = \frac{Z_1 + Z_2}{2}$$

El siguiente paso es encontrar el valor del trazo d. Este cálculo se puede hallar aplicando el teorema de Pitágoras para triángulos rectángulos de la siguiente manera:

$$d = \sqrt{(X_m - X_1)^2 + (Z_m - Z_1)^2}$$

Una vez hallado el valor del segmento llamado d, se procede a encontrar la pendiente de la recta. Este cálculo se puede hacer de la siguiente manera:

$$m = \frac{Z_m - Z_1}{X_m - X_1}$$

Para encontrar una recta perfectamente perpendicular a otra se debe aplicar la siguiente ecuación:

$$m_1 m_2 = -1$$

Entonces se tiene:

$$m_1 = \frac{-1}{m}$$

Teniendo la pendiente de la recta perpendicular, se puede calcular el valor de *q* de la siguiente manera:

$$q = \arctan(m_1)$$

Se procede a encontrar el valor del trazo Ra, que se puede encontrar de la siguiente manera:

$$R_a = \sqrt{R^2 + d^2}$$

Se procede a encontrar el punto medio del circulo de donde se desprende el segmento circular a dibujar (a,b). Esto se puede encontrar aplicando las siguientes ecuaciones:

$$a = X_m - R * \cos q$$

Y

$$b = Z_m - R * \sin q$$

Se procede a encontrar los valores de θ1 y θ2. Estos valores se encuentran la decidir desde y hasta dónde va el trazado del segmento circular. Para ello se pueden aplicar las siguientes ecuaciones:

$$\theta_1 = -Abs\frac{Arctan(Abs(b - Z_1))}{Arctan(Abs(a - X_1))}$$

Y

$$\theta_2 = -Abs\frac{Arctan(Abs(b - Z_2))}{Arctan(Abs(a - X_2))}$$

Entonces el intervalo queda de la siguiente forma:

$$dto = \frac{\theta_1 + \theta_2}{100}$$

Entonces se puede decir que el intervalo t va desde el θ1 hasta el θ2 cada valor dto. Una vez teniendo los intervalos se calculan los valores de X y Z de la siguiente forma:

$$X = R_a * \cos t + a$$

Y

$$Z = R_a * \sin t + b$$

- En caso de que la forma de interpolación sea circular y convexa, se hace los mismos cálculos al principio pero a la hora de hallar el punto (a,b) las ecuaciones cambien de la siguiente forma:

$$a = X_m + R * \cos q$$

Y

$$b = Z_m + R * \sin q$$

Las fórmulas para hallar los θ también varían y quedan así:

$$\theta_1 = Abs\,\frac{Arctan(Abs(b - Z_1))}{Arctan(Abs(a - X_1))}$$

Y

$$\theta_2 = Abs\frac{Arctan(Abs(b - Z_2))}{Arctan(Abs(a - X_2))}$$

Teniendo los mismos intervalos los nuevos valores de X y Z se pueden calcular de la siguiente manera:

$$X = -R_a * \cos t + a$$

Y

$$Z = -R_a * \sin t + b$$

El código desarrollado para el menú DIBUJO es el siguiente:

Figura 38. Código submenú DIBUJO

```
Private Sub dibujo_Click()
Label2.Caption = j
py = Picture1.ScaleHeight
esc = 30
For i = 2 To j
If valy(i) = 1 Then
Select Case valg(i)
Case 1: Picture1.Line (valx(i-1)*esc,py-valz(i-1)*esc)-(valx(i)*esc,py-valz(i)*esc)
```

```
Case 2: X1 = valx(i - 1)
        z1 = valz(i - 1)
        X2 = valx(i)
        z2 = valz(i)
        R = valr(i)
        xm = (X1 + X2) / 2
        zm = (z1 + z2) / 2
        d = Sqr((xm - X1) ^ 2 + (zm - z1) ^ 2)
        m = (zm - z1) / (xm - X1)
        m1 = -1 / m
        q = Atn(m1)
        Ra = Sqr(R ^ 2 + d ^ 2)
        a = xm - R * Cos(q)
        b = zm - R * Sin(q)
        o1 = -Abs(Atn((Abs(b - z1)) / (Abs(a - X1))))
        o2 = -Abs(Atn((Abs(b - z2)) / (Abs(a - X2))))
        dto = (o2 - o1) / 100
        't = o1: Do: o2
        For t = o1 To o2 Step dto
        x = Ra * Cos(t) + a
        z = Ra * Sin(t) + b
        Picture1.Line (x*esc,py-z*esc)-(x*esc+2,py-(z+1)*esc)
        Next t
Case 3: X1 = valx(i - 1)
        z1 = valz(i - 1)
        X2 = valx(i)
        z2 = valz(i)
        R = valr(i)
        xm = (X1 + X2) / 2
        zm = (z1 + z2) / 2
        d = Sqr((xm - X1) ^ 2 + (zm - z1) ^ 2)
        m = (zm - z1) / (xm - X1)
        m1 = -1 / m
        q = Atn(m1)
        Ra = Sqr(R ^ 2 + d ^ 2)
        a = xm + R * Cos(q)
        b = zm + R * Sin(q)
        'o1 = Abs(Atn((b - z1) / (a - X1)))
        o1 = Abs(Atn((Abs(b - z1)) / (Abs(a - X1))))
        o2 = Abs(Atn((Abs(b - z2)) / (Abs(a - X2))))
        dto = (o2 - o1) / 100
        't = o1: Do: o2
        For t = o1 To o2 Step dto
        x = -Ra * Cos(t) + a
        z = -Ra * Sin(t) + b
        Picture1.Line (x*esc,py-z*esc)-(x*esc+2,py-(z+1)*esc
        Next t
End Select
End If
Next i
End Sub
```

Fuente: Autores

7.2.2.2 Limpiar

El submenú LIMPIAR ejecuta un dibujo vacío dentro del área de dibujo con el fin de borrar el esquema creado previamente cuando sea necesaria esta acción.

El código encargado de generar este paso es:

Figura 39. Código submenú LIMPIAR

```
Private Sub limpiar_Click()
Picture1.Picture = Nothing
End Sub
```

Fuente: Autores.

7.2.3 Comunicaciones

Por medio de este menú se puede establecer la conexión entre el computador del usuario y el controlador lógico programable PLC. Se compone de los siguientes submenús:

7.2.3.1 Conectar

Este submenú es el encargado de reservar un canal de comunicación entre el computador y el PLC usando los comandos winsock del sistema operativo Windows.

El submenú CONECTAR se ejecuta de la siguiente manera:

- Verificar que no haya canales de comunicación abiertos entre el computador y el PLC. En caso de que si haya un canal abierto, se cierra inmediatamente y se habilita uno nuevo totalmente diferente.
- Se lee la dirección IP con la que se quiere intentar conectar.

- Arranca un temporizador que verifica si se ha hecho la conexión efectivamente o no.
- Si se logra hacer la conexión, se debe mantener hasta que el usuario indique lo contrario.

El código utilizado para este submenú es el siguiente:

Figura 40. Código submenú CONECTAR

```
Private Sub conectar_Click()
Dim StartTime
If (Winsock1.State <> sckClosed) Then
    Winsock1.Close
End If
Winsock1.RemoteHost = Text3.Text
Winsock1.Connect

StartTime = Timer

Do While ((Timer < StartTime + 2) And (Winsock1.State <> 7))
DoEvents
Loop
If (Winsock1.State = 7) Then
   Label1.Caption = "Connected"
   Label1.BackColor = &HFF00&
Else
   Label1.Caption = "Can't connect to " & Winsock1.RemoteHost
   Label1.BackColor = &HFF
End If
End Sub
```

Fuente: Autores.

7.2.3.2 Desconectar

La función de este submenú es romper la comunicación, establecida en el submenú CONECTAR, entre el computador del usuario y el controlador lógico programable para dejar libre el canal de comunicación.

El submenú DESCONECTAR se ejecuta de la siguiente manera:

- Si el canal de comunicación entre el computador y el PC está abierto, inmediatamente se cierra, y si se encuentra cerrado se mantiene de esa manera.
- Se espera hasta que el PLC reporte el cierre del canal de comunicación de forma correcta.
- Se le informa al usuario que el canal de comunicación ha sido cerrado exitosamente.

El código utilizado para este submenú es el siguiente:

Figura 41. Código submenú DESCONECTAR

```
Private Sub desconectar_Click()
If (Winsock1.State <> sckClosed) Then
    Winsock1.Close
End If
Do While (Winsock1.State <> sckClosed)
    DoEvents
Loop
Label1.Caption = "Disconnected"
Label1.BackColor = &HFF
End Sub
```

Fuente: Autores.

7.2.3.3 Leer

La función de este submenú, como su nombre lo indica, es leer las primeras 10 (diez) posiciones de datos tipo Word en el PLC, desde la dirección %MW0 hasta la dirección %MW9. Este tipo de comunicación se hace por medio del protocolo modbus TCP/IP.

El submenú LEER se ejecuta de la siguiente manera:

- Si el canal de comunicación está disponible, se envía una petición de lectura; si no se encuentra disponible se le informa al usuario que la comunicación ha sido interrumpida.
- Para hacer una petición de lectura por medio del protocolo modbus TCP/IP, se debe hacer una cadena de texto con la trama respectiva “0 0 0 0 0 6 1 3 0 0 0 15”.
- Se envía la trama a través del socket de comunicación.
- Se establecen las banderas de comunicación de forma adecuada.

El código utilizado para este submenú es el siguiente:

Figura 42. Código submenú LEER

```
Private Sub leer_Click()
If (Winsock1.State = 7) Then
MbusQuery=Chr(0)+Chr(0)+Chr(0)+Chr(0)+Chr(0)+Chr(6)+Chr(1)+Chr(3)+Chr(0)+Chr(0)+Chr(0)+Chr(15)
Winsock1.SendData MbusQuery
ModbusWait = True
ModbusTimeOut = 0
Timer1.Enabled = True
Else
MsgBox ("Device not connected via TCP/IP")
Timer2.Enabled = False
End If
End Sub
```

Fuente: Autores.

7.2.3.4 Escribir

Este submenú tiene como función, como su nombre lo indica, escribir los datos desde el computador del usuario a la memoria del controlador lógico programable PLC. Esta memoria está comprendida entre %MW0 hasta %MW9.

El submenú ESCRIBIR se ejecuta de la siguiente manera:

- Se establece la dirección IP en donde se van a escribir los datos transmitidos.
- Se crea el vector a enviar de datos escritos. Este es un vector de 10 (diez) posiciones donde se escriben las coordenadas de los puntos a unir así como su forma de interpolación.
- Se obtiene la parte alta y la parte baja de la dirección donde se pretende escribir.
- Se obtiene la parte alta y la parte baja del tamaño del vector que se pretende escribir en el PLC.
- Se calcula y se establece la cadena de comunicación.
- Se verifican las banderas de comunicación.
- Se envía la petición de escritura por medio del protocolo Modbus TCP/IP.
- Se espera el reporte de la cadena de datos recibida.

El código utilizado para este submenú es el siguiente:

Figura 43. Código Submenú ESCRIBIR

```
Private Sub escribir_Click()

Dim MbusWriteCommand As String
Dim StartLow As Byte
Dim StartHigh As Byte
Dim ByteLow As Byte
Dim ByteHigh As Byte
Dim i As Integer
Dim inicio As Integer
Dim longitud As Integer
Dim dato(10) As Integer
'inicializacion de las variables
inicio = 0
longitud = 10
'datos a escribir
pos = pos + 1
Label3.Caption = pos
escala = 111.11
dato(0) = valx(pos) * escala
dato(1) = valz(pos) * escala
dato(2) = 20
dato(3) = 120
dato(4) = 40
dato(5) = valy(pos)
dato(6) = 60
dato(7) = 70
dato(8) = 85
dato(9) = 95
dato(10) = 100


If (Winsock1.State = 7) Then
StartLow = inicio Mod 256
StartHigh = inicio \ 256
LengthLow = longitud Mod 256
LengthHigh = longitud \ 256

MbusWriteQuery = Chr(0) + Chr(0) + Chr(0) + Chr(0) + Chr(0) + Chr(7 + 2 * longitud)

+ Chr(1) + Chr(16) + Chr(StartHigh) + Chr(StartLow) + Chr(0)

+ Chr(longitud) + Chr(2 * longitud)
```

```
For i = 0 To Val(Text3.Text) - 1
    ByteLow = dato(i) Mod 256
    ByteHigh = dato(i) \ 256
    MbusWriteQuery = MbusWriteQuery + Chr(ByteHigh) + Chr(ByteLow)
Next i
MbusRead = False
MbusWrite = True
Winsock1.SendData MbusWriteQuery
ModbusWait = True
ModbusTimeOut = 0
Timer1.Enabled = True
Else
MsgBox ("Device not connected via TCP/IP")
End If
End Sub
```

Fuente: Autores.

7.2.3.5 Verificar

Permite leer un set de datos desde el PLC a intervalos regulares de tiempo (cada segundo) con el fin de revisar el estado de funcionamiento de la mesa posicionadora.

El código utilizado para este submenú es el siguiente:

Figura 44. Código submenú VERIFICAR

```
Private Sub verificar_Click()
'Label1.Caption = Now
Timer2.Enabled = True
End Sub
```

Fuente: Autores.

7.2.3.6 Cancelar

Como su nombre lo indica este submenú cancela la selección de submenús y vuelve al menú principal.

El código utilizado para este submenú es el siguiente:

Figura 45. Código submenú CANCELAR

```
Private Sub cancelar_Click()
Timer2.Enabled = False
End Sub
```

Fuente: Autores.

CONCLUSIONES

Se realiza programación en Visual Basic por medio de códigos G para control de mesa de dibujo comprobando que son un tipo de comandos que facilitan el control numérico.

Se comprueba que el controlador lógico programable es un dispositivo electrónico de fácil manejo y entendimiento esencial para la automatización de procesos en general.

Se genera un avance tecnológico de importancia en máquinas de control numérico al poder implementarse una mesa de dibujo que es capaz de plasmar figuras que posteriormente y por medio de investigación serán ejecutadas por máquinas de CNC.

Se usa módulo de expansión para el control de los transistores de motores paso a paso mejorando así su funcionamiento y velocidad.

•

BIBLIOGRAFIA

[1] http://es.scribd.com/doc/24979617/6/CARACTERISTICAS-DEL-CONTROL-NUMERICO-COMPUTARIZADO
Página revisada por última vez 28 de noviembre de 2013

[2]
http://www.elprisma.com/apuntes/ingenieria_mecanica/controlnumericocnc/
Página revisada por última vez el 16 de marzo de 2012
http://www.frlp.utn.edu.ar/mecanica/Materias/CNCMH/ClaseDemo.PDF
Página revisada por última vez el 16 de marzo de 2012

[3] http://r-luis.xbot.es/cnc/codes03.html
Página revisada por última vez el 16 de marzo de 2012

[4]http://www.monografias.com/trabajos17/motor-paso-a-paso/motor-paso-a-paso.shtml
Página revisada por última vez el 16 de marzo de 2012

[5] Controladores programables Twido, Guía de referencia de software, TWD USE 10AE spa Version 3.2

[6] Manual Twido Suite. Manual de introducción al uso de la herramienta de configuración, programación y depuración de controladores programables de la gama Twido. Creado: Instituto Schneider Electric de Formación, Fecha: 23 de Julio de 2008, Versión: 2.1, SCHNEIDER ELECTRIC ESPAÑA.

[7] Manual Modicon TM2. Módulos de E/S digitales. Guía de hardware, 06/2011.
[8] Autómatas programables Twido, guía de instalación del software, TWD USE 10AS spa Version 2.0.

Figura 1: http://www.knuth.de/produkt,25296,sprache,4.html
Página revisada por última vez el 16 de marzo de 2012

Figura 2: http://www.gratis-tutoriales.com.ar/tornos-cnc/fresadora-cnc-roland-comercial
Página revisada por última vez el 16 de marzo de 2012

ANEXOS

ANEXO A. MODULO DE SALIDAS DE TRANSISTOR DIGITALES

TM2DDO8TT

discrete output module M238 - 8 outputs 24 V
transistor - 1 screw terminal block

Main

Range of product	Modicon M238 logic controller
Product or component type	Discrete output module
Discrete output number	8
Discrete output type	Transistor
Discrete output voltage	24 V
Discrete output logic	Source
Discrete output current	0.5 A

Complementary

Range compatibility	Advantys OTB Twido
Output voltage limits	20.4...28.8 V
Current per channel	0.6 A
Current per output common	4 A
Number of common point	1
Response time	450 µs from state 0 to state 1 450 µs from state 1 to state 0
[Ures] residual voltage	≤ 0.4 V at state 1
Leakage current	0.1 mA
Inductive load	≤ 10 mH
Tungsten load	≤ 12 W
Short-circuit protection	With automatic reactivaton
Overload protection	With automatic reactivaton
Isolation between channels	None
Isolation between channels and internal logic	500 V for 1 minute

Current consumption	10 mA 5 V DC at state 1 for all output 20 mA 24 V DC at state 1 for all output
Local signalling	1 display block
Electrical connection	1 removable screw terminal block
Mounting support	35 mm symmetrical DIN rail
Product weight	0.085 kg

Environment

RoHS EUR conformity date	0830
RoHS EUR status	Compliant

Fuente: Manual Modicon TM2. Módulos de E/S digitales. Guía de hardware, 06/2011.

ANEXO B. CONTROLADOR LÓGICO PROGRAMABLE

Acerca de Twido

Introducción

El controlador Twido está disponible en dos modelos:
- Compacto
- Modular

El controlador compacto se encuentra disponible con:
- 10 E/S
- 16 E/S
- 24 E/S
- 40 E/S

El controlador modular se encuentra disponible con:
- 20 E/S
- 40 E/S

Es posible añadir E/S adicionales al controlador mediante módulos de E/S de ampliación. Puede haber:
- 15 módulos de ampliación de las E/S digitales o tipo de relé
- 8 módulos de ampliación del tipo de E/S analógicas

La conexión a un módulo de interfase del bus AS-Interface también permite administrar hasta 62 equipos slaves. Se trata del siguiente módulo:
- Módulo master de interfase del bus AS-Interface V2: TWDNOI10M3

Los controladores de base compacta de 24y 40 de E/S y todos los controladores modulares se pueden conectar a un módulo de interfase del bus de campo CANopen que le permite dirigir hasta 16 dispositivos slave CANopen (sin exceder los 16 Transmit-PDOs (TPDO) y 16 Receive-PDOs (RPDO)). Utilice el siguiente módulo:
- Módulo master de interfase del bus de campo CANopen: TWDNCO1M.

Existen varias opciones que pueden agregarse a los controladores base:
- Cartuchos de memoria
- Cartucho de reloj de tiempo real (RTC)
- Adaptadores de comunicaciones
- Módulos de ampliación de comunicaciones (sólo para controladores modulares)
- Módulo de interfase Ethernet (todos los controladores modulares y compactos, excepto TWDLCAE40DRF con interfase Ethernet integrada)
- Módulo de monitor de operación (sólo para controladores compactos)
- Módulo de ampliación de monitor de operación (sólo para controladores modulares)
- Simuladores de entradas (sólo controlador compacto)
- Cables de programación
- Cables de E/S digitales
- Sistemas precableados Telefast® con interfases de E/S

Los controladores base compactos de las series TWDLCAA40DRF y TWDLCAE40DRF integran funciones avanzadas:
- Puerto de red Ethernet 100Base-TX integrado: sólo para TWDLCAE40DRF
- Reloj de tiempo real (RTC) integrado: TWDLCAA40DRF y TWDLCAE40DRF
- Un cuarto contador rápido (FC): TWDLCAA40DRF y TWDLCAE40DRF
- Soporte de batería externa: TWDLCAA40DRF y TWDLCAE40DRF

Nombre del controlador	Referencia	Canales	Tipo de canal	Tipo de entrada/salida	Fuente de alimentación
Compacto de 40 E/S	TWDLCAE40DRF	24 16	Entradas Salidas	24 VDC Relé X 14 Transistores X 2 Puerto Ethernet	100/240 VAC

Funciones principales de los controladores

Introducción

De forma predeterminada, todas las E/S de los controladores se configuran como E/S binarias. Sin embargo, algunas E/S se pueden asignar a tareas específicas durante la configuración, como:

- Entrada RUN/STOP
- Entradas con retención
- Contadores rápidos:
  - Contadores progresivos/regresivos individuales: 5 kHz (de una fase)
  - Contadores muy rápidos: contadores progesivos/regresivos: 20 kHz (de dos fases)
- Salida de estado del controlador
- PWM (Modulación de ancho de pulso)
- Salida del generador de pulsos (PLS)

Los controladores Twido están programados con TwidoSoft, que permite utilizar las funciones siguientes:

- PWM
- PLS
- Contadores rápidos y muy rápidos
- PID y autoafinado del PID

Descripción general de las comunicaciones

Introducción

Los controladores Twido disponen de un puerto serie, o de un segundo puerto opcional, que se utiliza para servicios en tiempo real o de administración de sistemas. Los servicios en tiempo real proporcionan funciones de distribución de datos para intercambiar datos con dispositivos de E/S, así como funciones de administración para comunicarse con dispositivos externos. Los servicios de administración de sistemas controlan y configuran el controlador por medio de TwidoSoft. Cada puerto serie se utiliza para cualquiera de estos servicios, pero sólo el puerto serie 1 es válido para comunicarse con TwidoSoft.
Para poder utilizar estos servicios, existen tres protocolos disponibles en cada controlador:

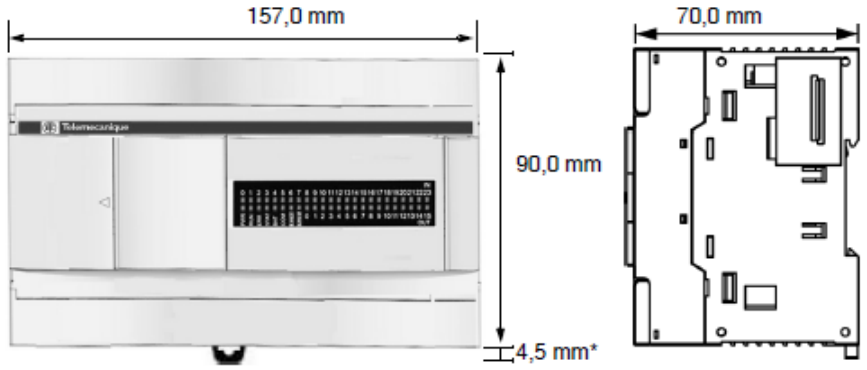
- Conexión remota
- Modbus
- ASCII

Además, el controlador compacto TWDLCAE40DRF proporciona un puerto de comunicación RJ45 Ethernet integrado que permite llevar a cabo todas las tareas de comunicación en tiempo real y de administración del sistema a través de la red. Las comunicaciones Ethernet implementan el siguiente protocolo:

- TCP/IP Modbus

TWDLCA•40-DRF

Los siguientes diagramas muestran las dimensiones del controlador compacto de la serie TWDLCA•40DRF.

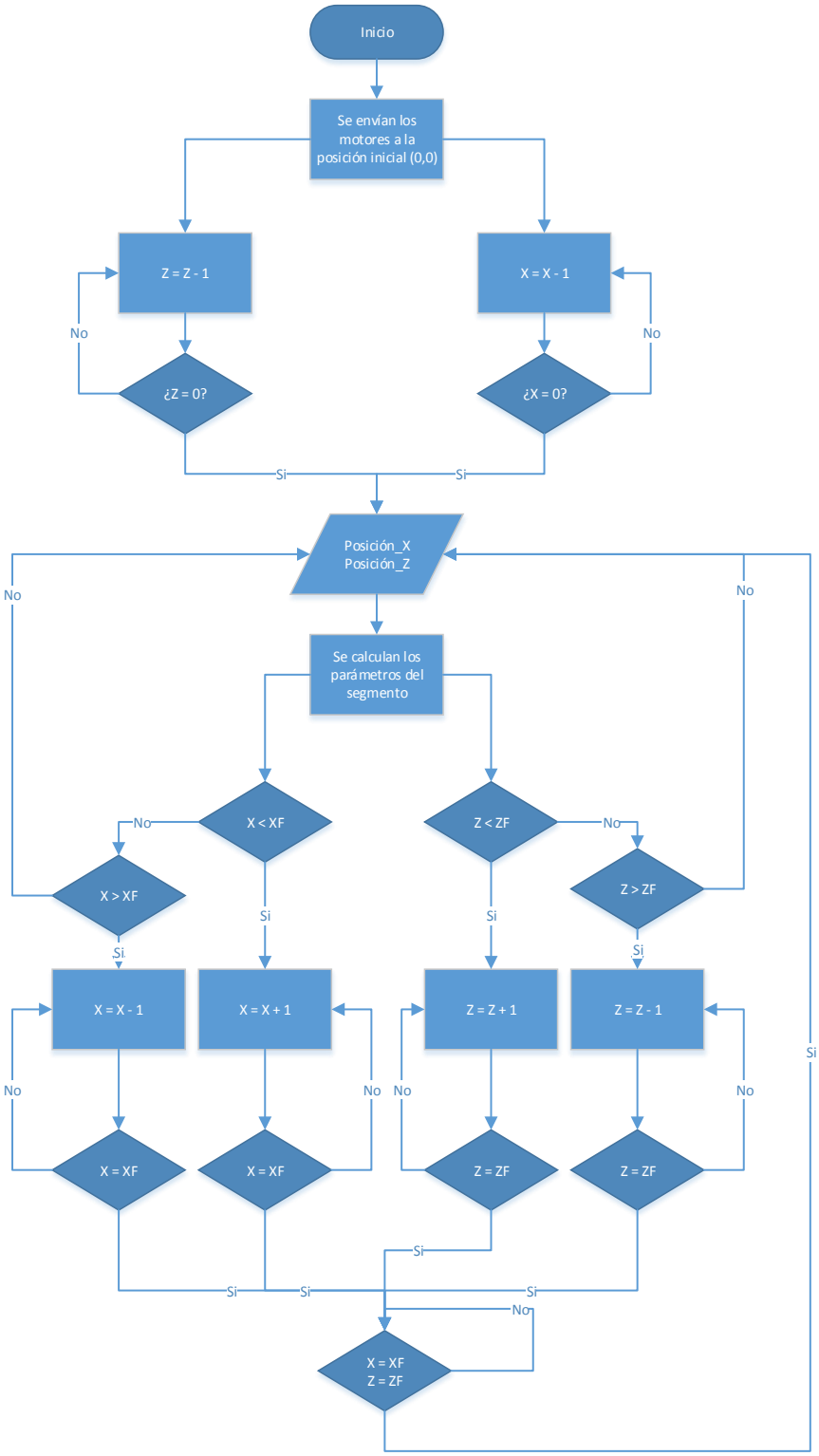


Nota: * 8,5 mm cuando se retira la abrazadera.

Tipo de controlador	Ilustración
Controladores compactos de 40 E/S: A continuación se detallan las funciones compartidas por controladores de las series TWDLCAA40DRF y TWDLCAE40DRF: ● 24 entradas digitales, 14 de relé y 2 salidas de transistor ● 2 potenciómetros analógicos ● 1 puerto serie integrado ● 1 slot para un puerto serie adicional ● RTC integrado ● Compartimiento de batería para batería externa reemplazable por el usuario ● Admite hasta 7 módulos de ampliación de E/S. ● Admite hasta dos módulos de interfase del bus AS-Interface V2 ● Admite un módulo master de interfase del bus de campo CANopen: ● Admite un cartucho de memoria opcional (de 32 ó 64 KB) ● Admite un módulo de monitor de operación opcional Características específicas de TWDLCAA40DRF: ● Admite un módulo de interfase Ethernet ConneXium TwidoPort Características específicas de TWDLCAE40DRF: ● Puerto RJ45 de interfase Ethernet integrado	TWDLCAA40DRF  TWDLCAE40DRF 

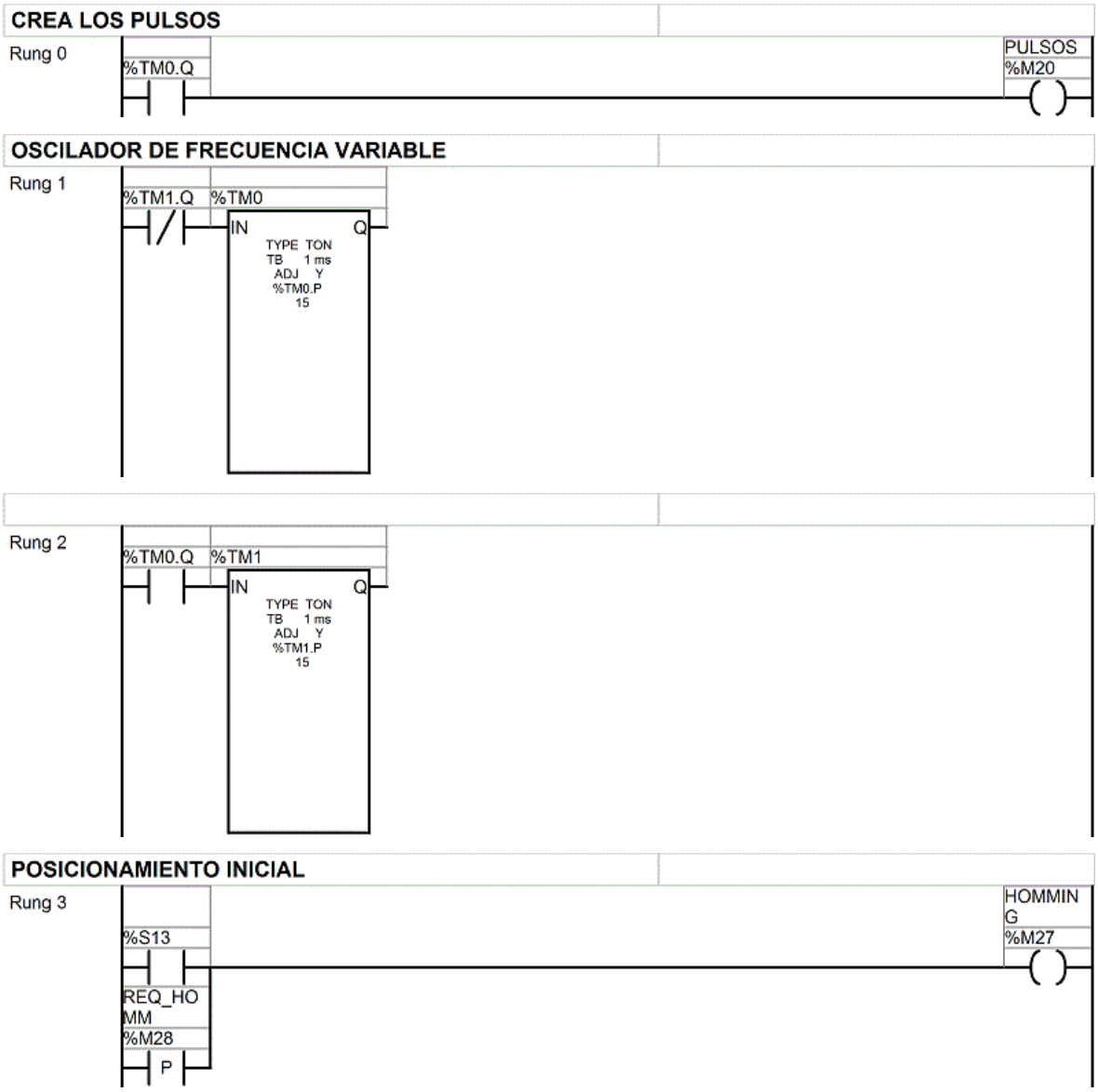
Fuente: Controladores programables Twido Guía de referencia de hardware TWD USE 10AS spa Versión 3.2

ANEXO C. DIAGRAMA DE FLUJO PROGRAMA PLC

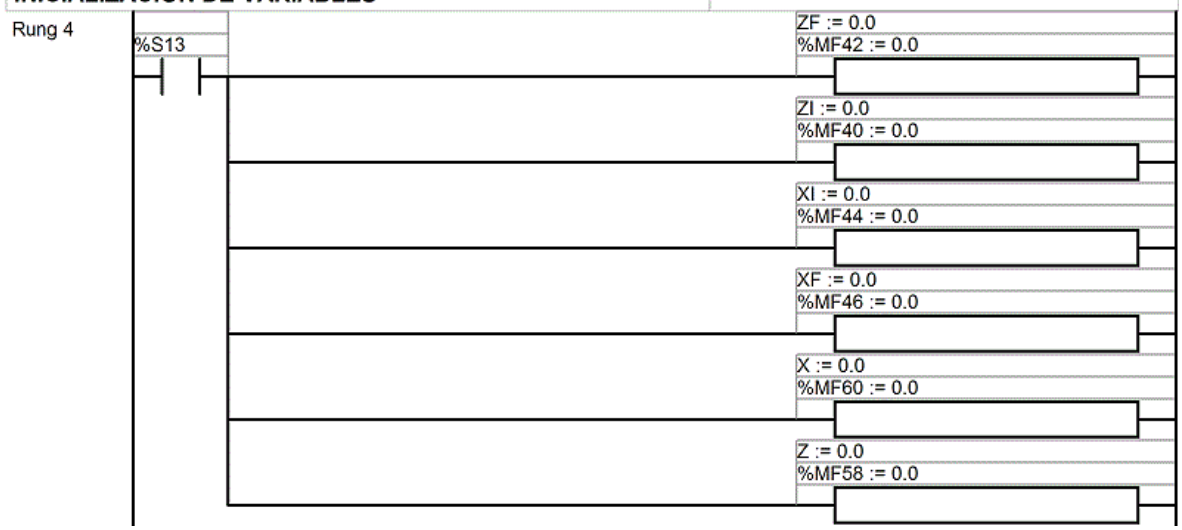


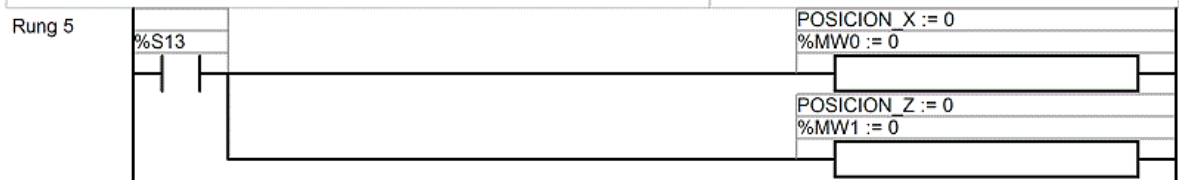
Fuente: Autores

ANEXO D. PROGRAMA PLC TWIDO TELEMECANIQUE

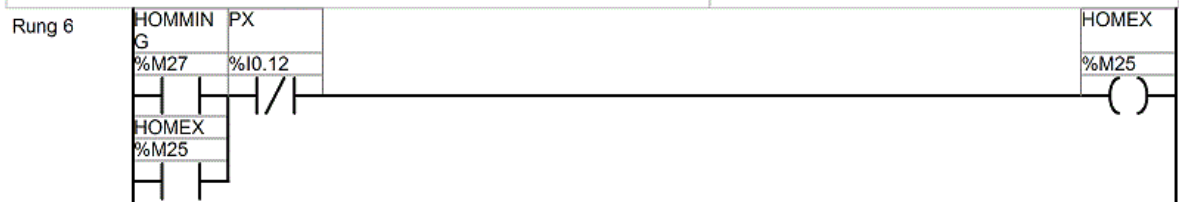


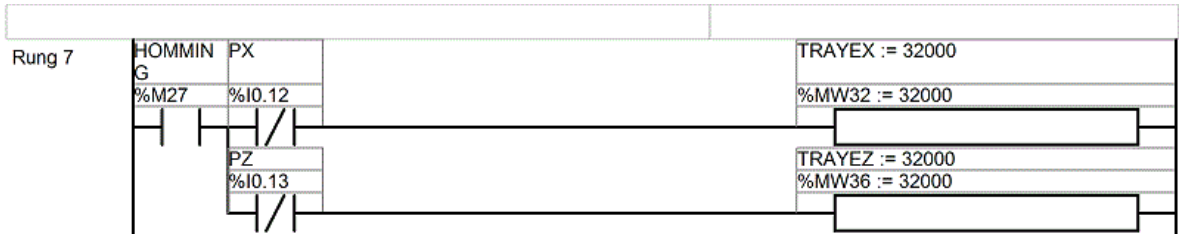
INICIALIZACION DE VARIABLES

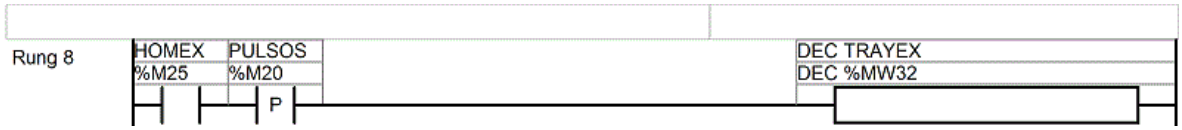


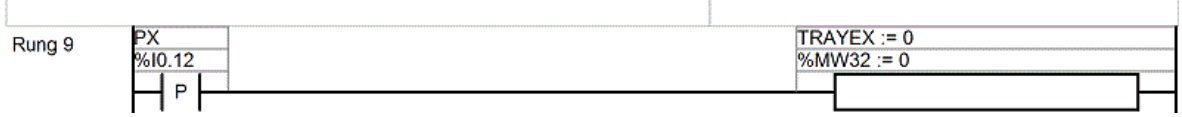


POSICION INICIAL EN EL EJE X



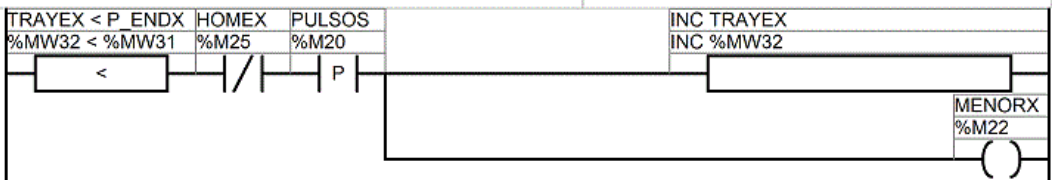






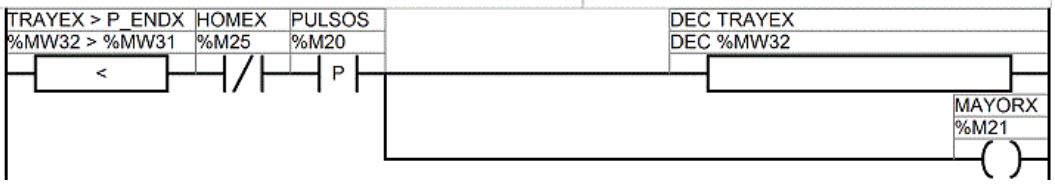
CHEQUEA SI LA POSICION FINAL ES MENOR QUE LA ACTUAL

Rung 10



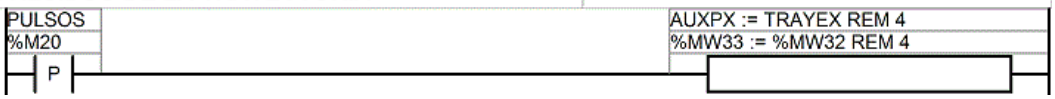
CHEQUEA SI LA POSICION FINAL ES MAYOR QUE LA ACTUAL

Rung 11

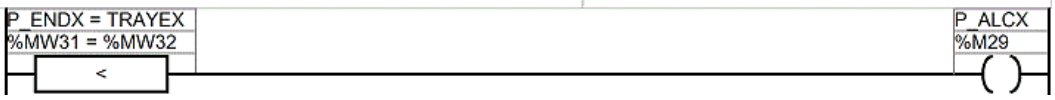


HALLA EL RESIDUO DE LA TRAYECTORIA

Rung 12

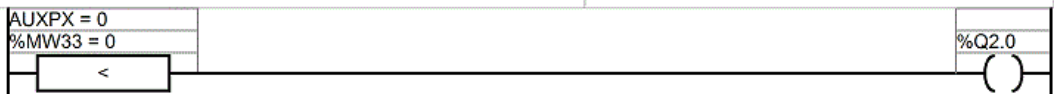


Rung 13

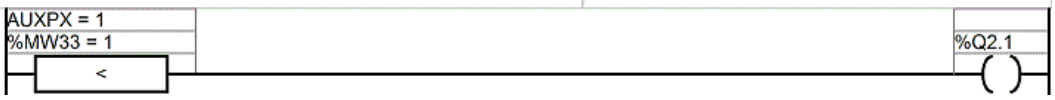


HABILITA LAS BOBINAS

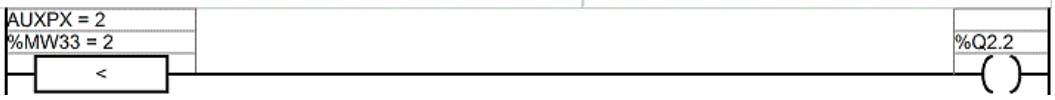
Rung 14



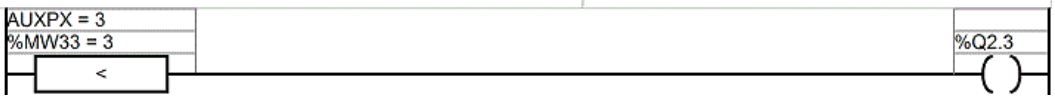
Rung 15



Rung 16

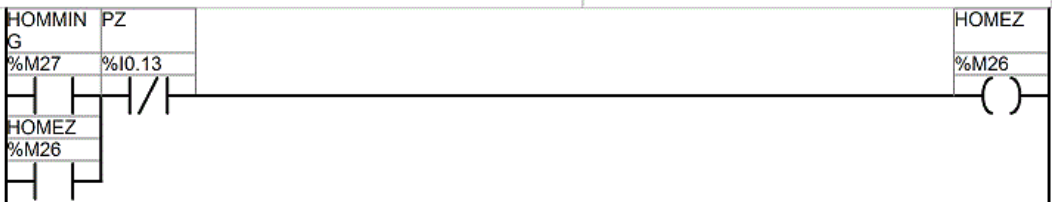


Rung 17

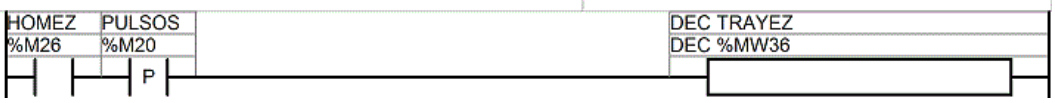


POSICIONAMIENTO EN EL EJE Z

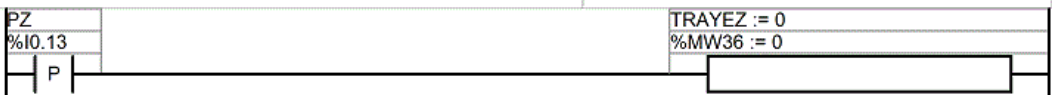
Rung 18



Rung 19

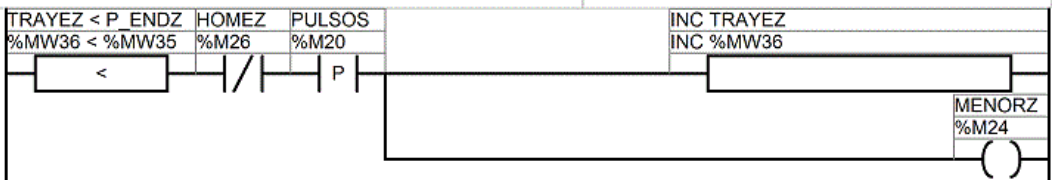


Rung 20



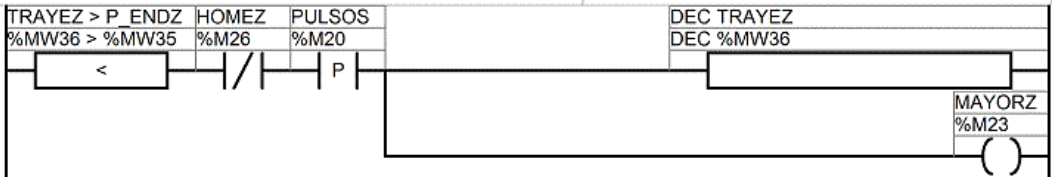
CHEQUEA SI LA POSICION FINAL ES MENOR QUE LA ACTUAL

Rung 21



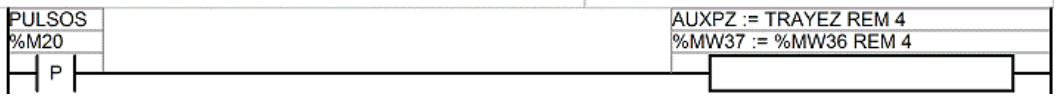
CHEQUEA SI LA POSICION FINAL ES MAYOR QUE LA ACTUAL

Rung 22

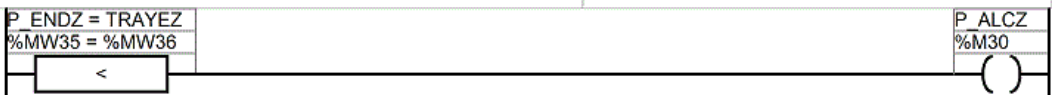


HALLA EL RESIDUO DE LA TRAYECTORIA

Rung 23

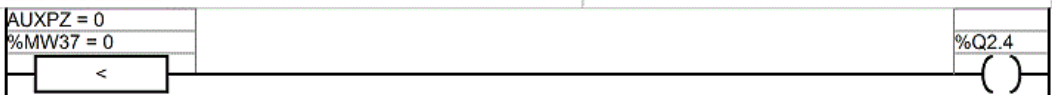


Rung 24



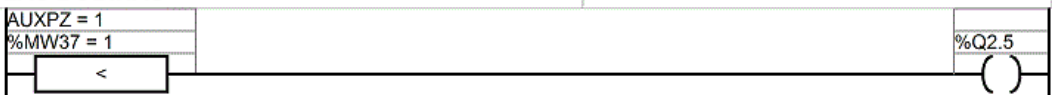
HABILITA LAS BOBINAS

Rung 25



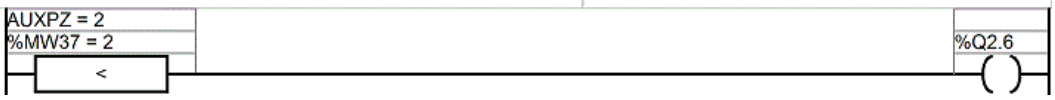
HABILITA LAS BOBINAS

Rung 26



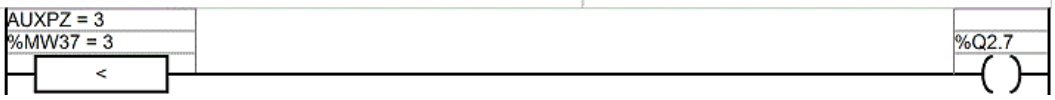
HABILITA LAS BOBINAS

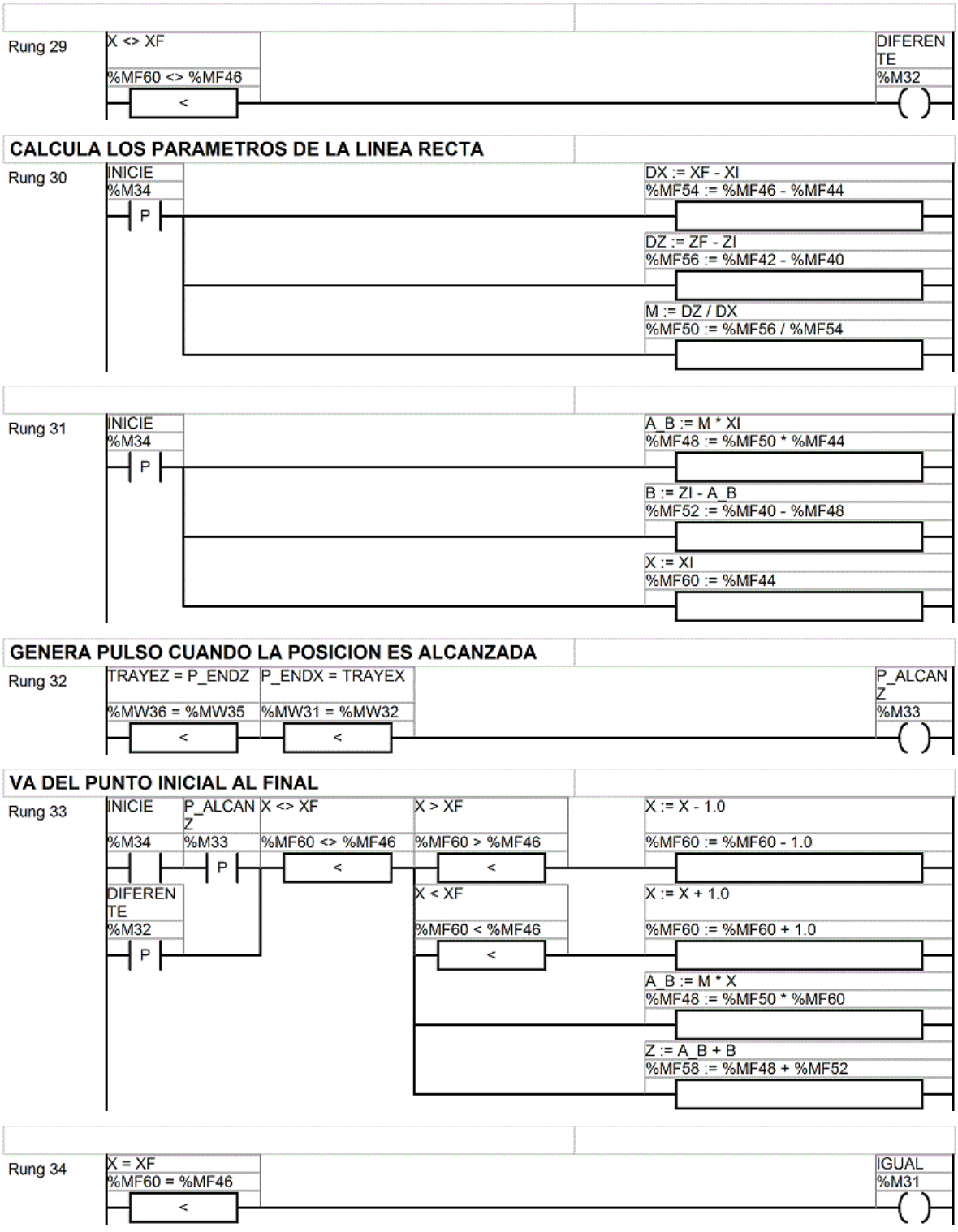
Rung 27

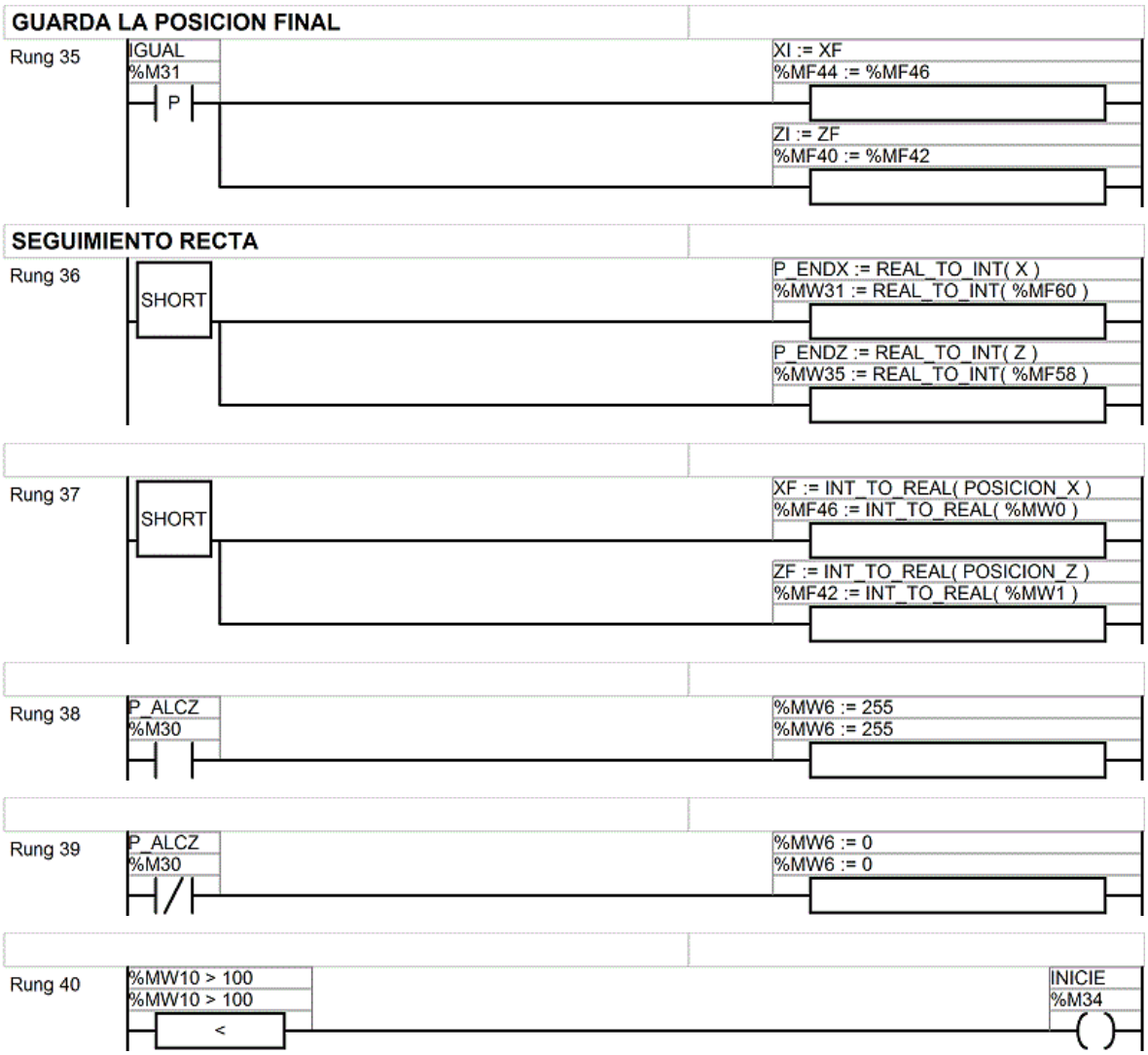


HABILITA LAS BOBINAS

Rung 28



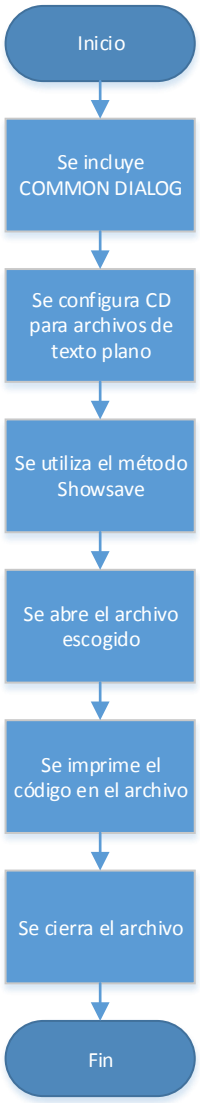

Rung 29
X <> XF
%MF60 <> %MF46
<
DIFEREN
TE
%M32
CALCULA LOS PARAMETROS DE LA LINEA RECTA
Rung 30
INICIE
%M34
P
DX := XF - XI
%MF54 := %MF46 - %MF44
DZ := ZF - ZI
%MF56 := %MF42 - %MF40
M := DZ / DX
%MF50 := %MF56 / %MF54
Rung 31
INICIE
%M34
P
A_B := M * XI
%MF48 := %MF50 * %MF44
B := ZI - A_B
%MF52 := %MF40 - %MF48
X := XI
%MF60 := %MF44
GENERA PULSO CUANDO LA POSICION ES ALCANZADA
Rung 32
TRAYEZ = P_ENDZ
%MW36 = %MW35
<
P_ENDX = TRAYEX
%MW31 = %MW32
<
P_ALCAN
Z
%M33
VA DEL PUNTO INICIAL AL FINAL
Rung 33
INICIE
%M34
P_ALCAN
Z
%M33
P
DIFEREN
TE
%M32
P
X <> XF
%MF60 <> %MF46
<
X > XF
%MF60 > %MF46
<
X < XF
%MF60 < %MF46
<
X := X - 1.0
%MF60 := %MF60 - 1.0
X := X + 1.0
%MF60 := %MF60 + 1.0
A_B := M * X
%MF48 := %MF50 * %MF60
Z := A_B + B
%MF58 := %MF48 + %MF52
Rung 34
X = XF
%MF60 = %MF46
<
IGUAL
%M31



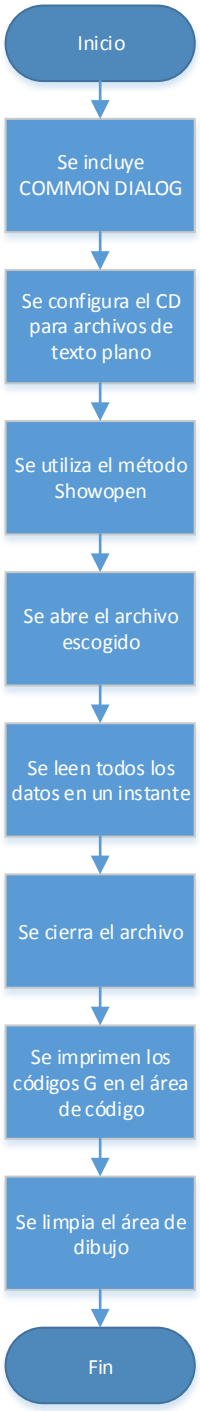
Fuente: Autores.

ANEXO E. DIAGRAMA DE FLUJO PROGRAMA VISUAL BASIC

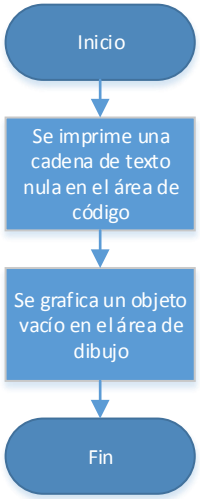
- Diagrama de flujo submenú GUARDAR:



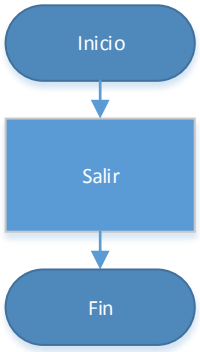
- Diagrama de flujo submenú CARGAR:



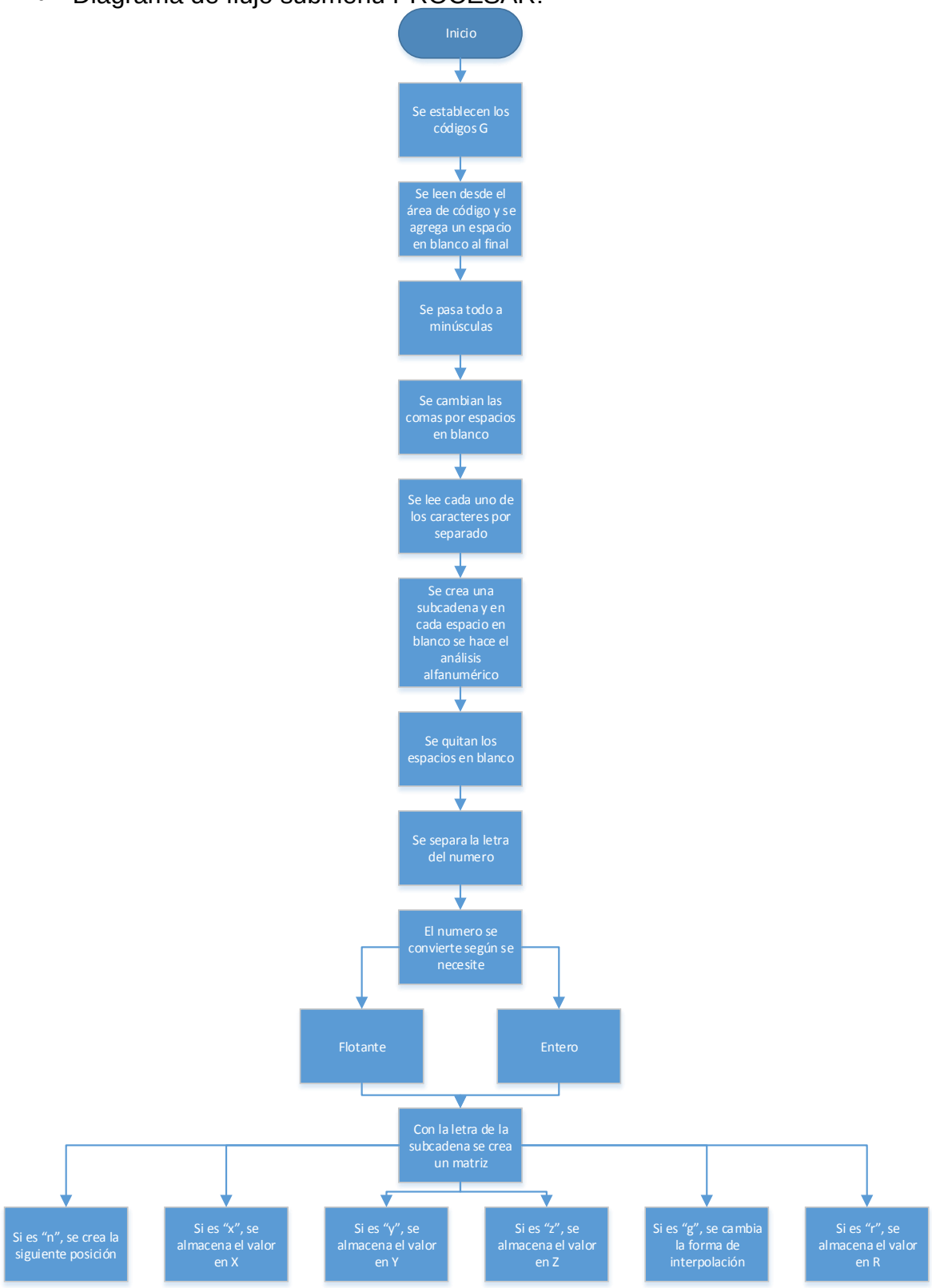
- Diagrama de flujo submenú NUEVO:



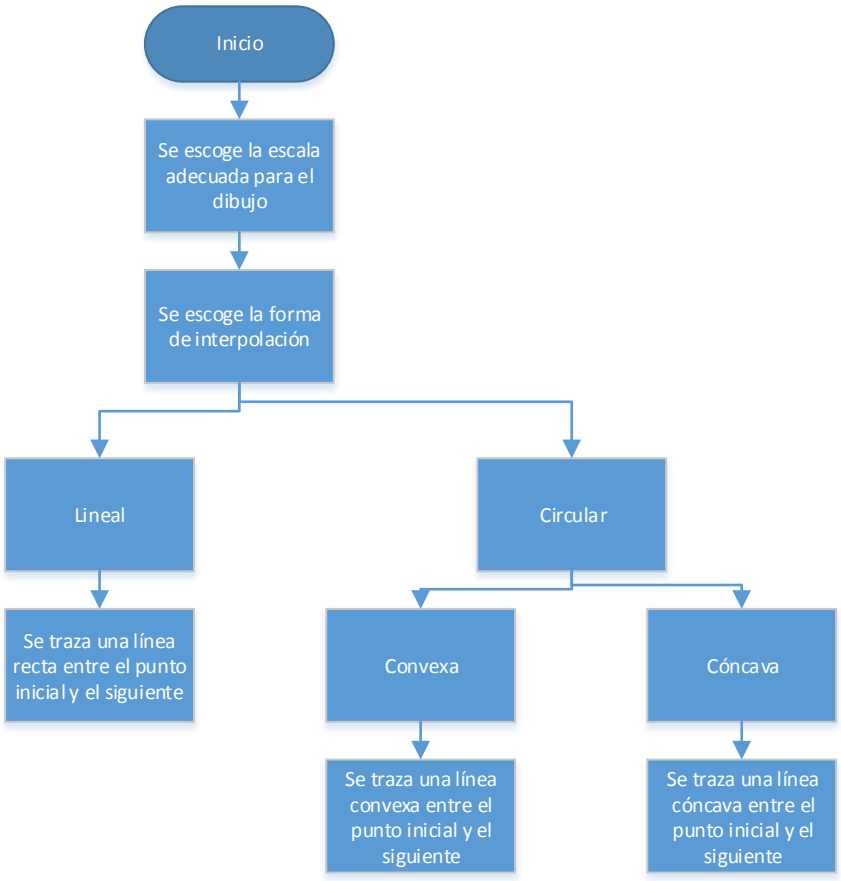
- Diagrama de flujo submenú SALIR:



- Diagrama de flujo submenú PROCESAR:

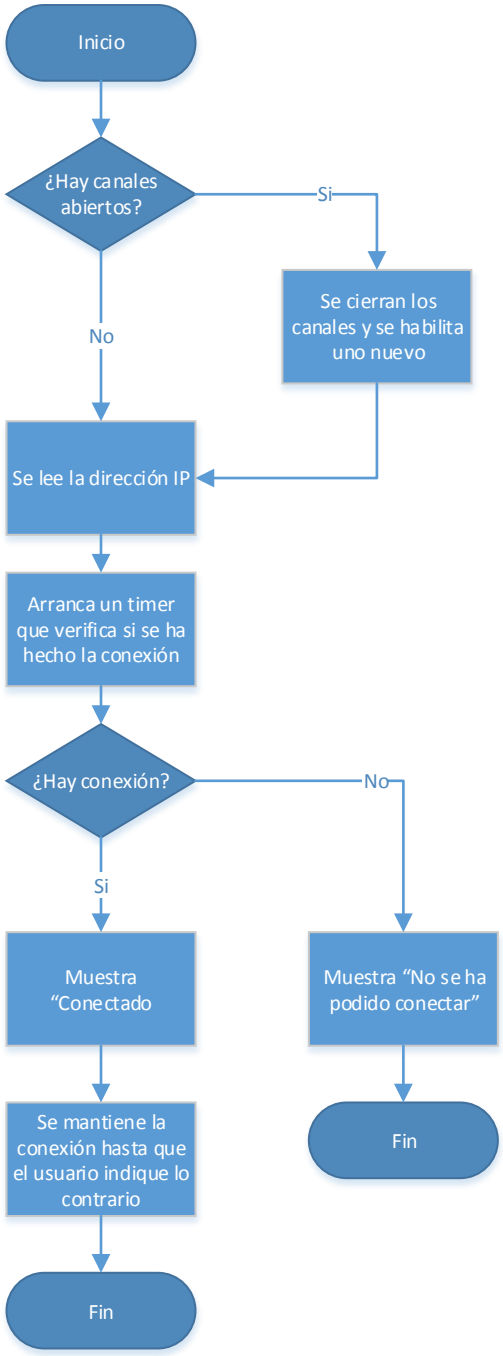


- Diagrama de flujo submenú DIBUJO:

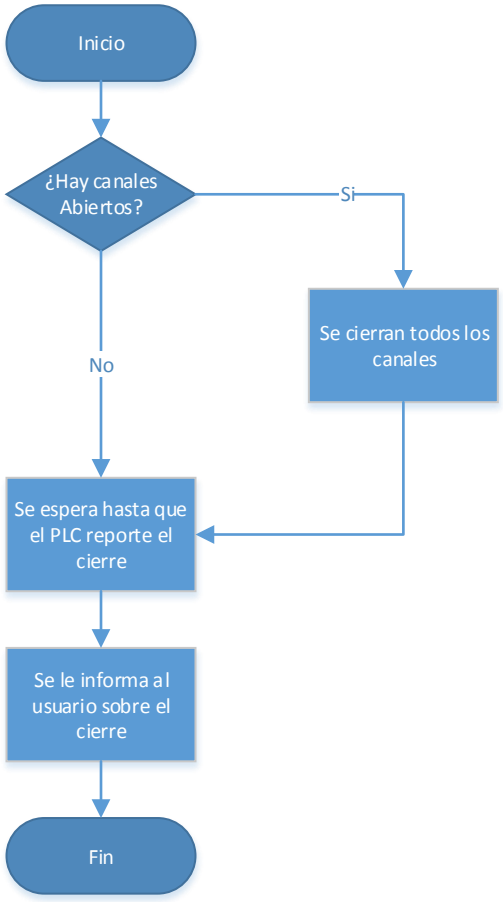


- Diagrama de flujo submenú LIMPIAR:

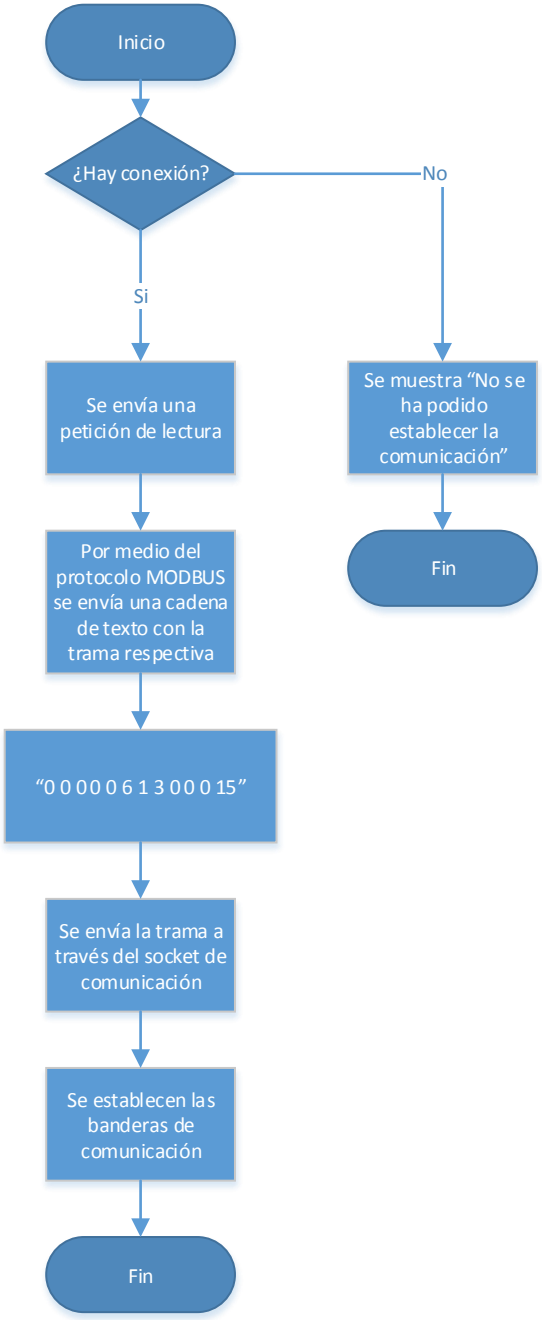
- Diagrama de flujo submenú CONECTAR:



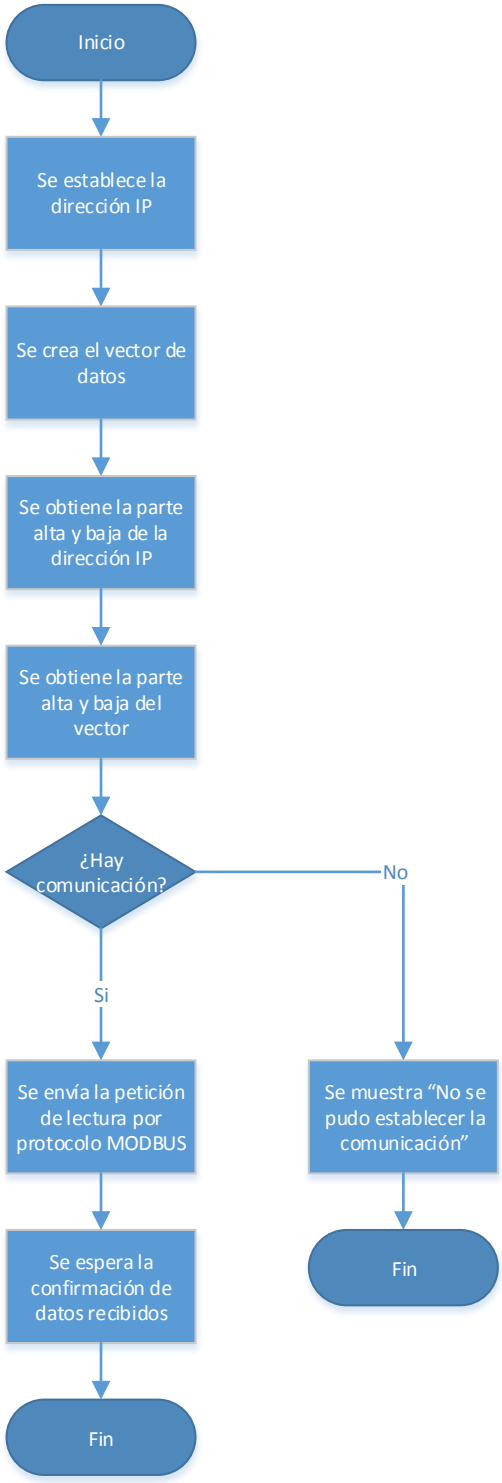
- Diagrama de flujo submenú DESCONECTAR:



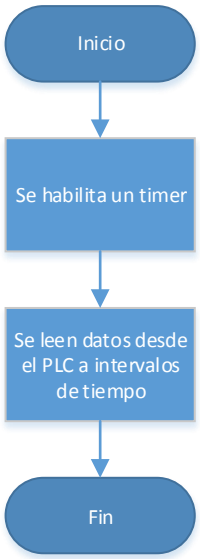
- Diagrama de flujo submenú LEER:



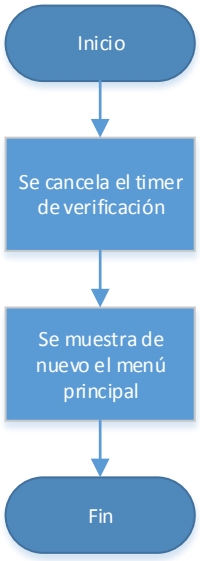
- Diagrama de flujo submenú ESCRIBIR:



- Diagrama de flujo submenú VERIFICAR:



- Diagrama de flujo submenú CANCELAR:



ANEXO F. PROGRAMA VISUAL BASIC

```
Dim valx(100), valy(100), valz(100), valg(100), valr(100), pos As Integer
Dim j As Integer

Private Sub cancelar_Click()
Timer2.Enabled = False
End Sub

Private Sub cargar_Click()
Dim VarTexto As String
CD1.Filter = "Ficheros de Texto |*.txt"
CD1.ShowOpen
Open CD1.FileName For Input As #1
Contenido = Input(LOF(1), #1)        'leemos todo el fichero de un golpe
Close #1
'TBTexto = VarTexto
Text1.Text = Contenido
Picture1.Picture = Nothing
End Sub

Private Sub conectar_Click()
Dim StartTime
If (Winsock1.State <> sckClosed) Then
   Winsock1.Close
End If
Winsock1.RemoteHost = Text3.Text
Winsock1.Connect
```

```
StartTime = Timer

Do While ((Timer < StartTime + 2) And (Winsock1.State <> 7))
DoEvents
Loop
If (Winsock1.State = 7) Then
  Label1.Caption = "Connected"
  Label1.BackColor = &HFF00&
Else
  Label1.Caption = "Can't connect to " & Winsock1.RemoteHost
  Label1.BackColor = &HFF
End If
End Sub

Private Sub desconectar_Click()
If (Winsock1.State <> sckClosed) Then
   Winsock1.Close
End If
Do While (Winsock1.State <> sckClosed)
   DoEvents
Loop
Label1.Caption = "Disconnected"
Label1.BackColor = &HFF
End Sub

Private Sub dibujo_Click()
Label2.Caption = j
py = Picture1.ScaleHeight
esc = 30
```

```
For i = 2 To j
If valy(i) = 1 Then
Select Case valg(i)
Case 1: Picture1.Line (valx(i - 1) * esc, py - valz(i - 1) * esc)-(valx(i) * esc, py - valz(i)
* esc)
Case 2: X1 = valx(i - 1)
    z1 = valz(i - 1)
    X2 = valx(i)
    z2 = valz(i)
    R = valr(i)
    xm = (X1 + X2) / 2
    zm = (z1 + z2) / 2
    d = Sqr((xm - X1) ^ 2 + (zm - z1) ^ 2)
    m = (zm - z1) / (xm - X1)
    m1 = -1 / m
    q = Atn(m1)
    Ra = Sqr(R ^ 2 + d ^ 2)
    a = xm - R * Cos(q)
    b = zm - R * Sin(q)
    o1 = Atn((-b + z1) / (-a + X1))
    o2 = Atn((-b + z2) / (-a + X2))
    'o1 = -Abs(Atn((b - z1) / (a - X1)))
    'o2 = -Abs(Atn((b - z2) / (a - X2)))
    dto = (o2 - o1) / 100
    't = o1: Do: o2
    For t = o1 To o2 Step dto
    x = Ra * Cos(t) + a
    z = Ra * Sin(t) + b
    Picture1.Line (x * esc, py - z * esc)-(x * esc + 2, py - (z + 1) * esc)
    Next t
```

```
Case 3: X1 = valx(i - 1)
        z1 = valz(i - 1)
        X2 = valx(i)
        z2 = valz(i)
        R = valr(i)
        xm = (X1 + X2) / 2
        zm = (z1 + z2) / 2
        d = Sqr((xm - X1) ^ 2 + (zm - z1) ^ 2)
        m = (zm - z1) / (xm - X1)
        m1 = -1 / m
        q = Atn(m1)
        Ra = Sqr(R ^ 2 + d ^ 2)
        a = xm + R * Cos(q)
        b = zm + R * Sin(q)
        o1 = Atn((-b + z1) / (-a + X1))
        o2 = Atn((-b + z2) / (-a + X2))
        'o1 = Abs(Atn((b - z1) / (a - X1)))
        'o2 = Abs(Atn((b - z2) / (a - X2)))
        dto = (o2 - o1) / 100
        't = o1: Do: o2
        For t = o1 To o2 Step dto
        x = -Ra * Cos(t) + a
        z = -Ra * Sin(t) + b
        Picture1.Line (x * esc, py - z * esc)-(x * esc + 2, py - (z + 1) * esc)
        Next t
End Select
End If
Next i
End Sub
```

```
Private Sub escribir_Click()

Dim MbusWriteCommand As String
Dim StartLow As Byte
Dim StartHigh As Byte
Dim ByteLow As Byte
Dim ByteHigh As Byte
Dim i As Integer
Dim inicio As Integer
Dim longitud As Integer
Dim dato(10) As Integer
'inicializacion de las variables
inicio = 0
longitud = 10
'datos a escribir
pos = pos + 1
Label3.Caption = pos
escala = 111.11
dato(0) = valx(pos) * escala
dato(1) = valz(pos) * escala
dato(3) = 120
dato(4) = 40
dato(5) = valy(pos)
dato(6) = 60
dato(7) = 70
dato(8) = 85
dato(9) = 95
dato(10) = 100

If (Winsock1.State = 7) Then
```

```
StartLow = inicio Mod 256
StartHigh = inicio \ 256
LengthLow = longitud Mod 256
LengthHigh = longitud \ 256

MbusWriteQuery = Chr(0) + Chr(0) + Chr(0) + Chr(0) + Chr(0) + Chr(7 + 2 *
longitud) + Chr(1) + Chr(16) + Chr(StartHigh) + Chr(StartLow) + Chr(0) +
Chr(longitud) + Chr(2 * longitud)
For i = 0 To Val(Text3.Text) - 1
   ByteLow = dato(i) Mod 256
   ByteHigh = dato(i) \ 256
   MbusWriteQuery = MbusWriteQuery + Chr(ByteHigh) + Chr(ByteLow)
Next i
MbusRead = False
MbusWrite = True
Winsock1.SendData MbusWriteQuery
ModbusWait = True
ModbusTimeOut = 0
Timer1.Enabled = True
Else
MsgBox ("Device not connected via TCP/IP")
End If
End Sub

Private Sub guardar_Click()
Dim VarTexto As String
Contenido = Text1.Text
CD1.Filter = "Ficheros de Texto |*.txt"
CD1.ShowSave
Open CD1.FileName For Output As #1
```

```
Print #1, Contenido
Close #1

End Sub
Private Sub limpiar_Click()
Picture1.Picture = Nothing
End Sub

Private Sub nuevo_Click()
Text1.Text = ""
Picture1.Picture = Nothing
End Sub

Private Sub procesar_Click()
Dim comandos, subcom As String
Dim i As Integer
j = 0
g = 1
y = 0
comandos = Text1.Text & " "
comandos = LCase(comandos)
comandos = Replace(comandos, vbNewLine, " ", , , vbTextCompare)
comandos = Replace(comandos, ",", " ", , , vbTextCompare)
subcom = ""
For i = 1 To Len(comandos)
   car = Mid$(comandos, i, 1)
   If car <> " " Then
      subcom = subcom & car
      Else
      subcom = Trim(subcom)
```

```
        letra = Left$(subcom, 1)
        If Len(subcom) > 0 Then
           numero = Right$(subcom, Len(subcom) - 1)
        End If
        cumint = Val(numero)
        Select Case letra
        Case "n": j = j + 1
                valg(j) = g
                valy(j) = y
                valr(j) = 0
        Case "x": valx(j) = cumint
        Case "z": valz(j) = cumint
        Case "y": valy(j) = cumint
                y = cumint
        Case "g": g = cumint
                valg(j) = g
        Case "r": valr(j) = cumint
        End Select
        subcom = ""
        End If

Next i
comandos = ""
For i = 1 To j
   comandos = comandos & "g=" & valg(i) & " x=" & valx(i) & " y=" & valy(i) & "
z=" & valz(i) & " r=" & valr(i) & vbNewLine
   Next i
   Text2.Text = comandos
End Sub
```

```
Private Sub salir_Click()
End
End Sub

Private Sub Timer1_Timer()
Dim StartTime
If (Winsock1.State <> sckClosed) Then
   Winsock1.Close
End If
'Winsock1.RemoteHost = Text1.Text
Winsock1.Connect

StartTime = Timer

Do While ((Timer < StartTime + 2) And (Winsock1.State <> 7))
DoEvents
Loop
If (Winsock1.State = 7) Then
  Label1.Caption = "Connected"
  Label1.BackColor = &HFF00&
Else
  Label1.Caption = "Can't connect to " & Winsock1.RemoteHost
  Label1.BackColor = &HFF
End If
End Sub

Private Sub Timer2_Timer()
'Label1.Caption = Now
caminar
End Sub
```

```
Private Sub verificar_Click()
'Label1.Caption = Now
Timer2.Enabled = True
End Sub

Private Sub Winsock1_DataArrival(ByVal datalength As Long)
Dim b As Byte
Dim j As Byte
Dim dato(100) As Byte
Dim valors As String
Dim numeros(100) As Integer

'Label1.Caption = "datos recividos"
For i = 1 To datalength
   Winsock1.GetData b
   dato(i) = b
   valors = valors & dato(i) & " "
Next
'Label1.Caption = valors
j = 1
Label1.Caption = ""
For i = 10 To datalength Step 2
numeros(j) = dato(i) * 256 + dato(i + 1)
Label1.Caption = Label1.Caption & numeros(j) & " "
j = j + 1
Next
'For i = 1 To j
'Label1.Caption = Label1.Caption & numeros(j) & " "
'Next i
```

```
End Sub
Private Sub leer_Click()
If (Winsock1.State = 7) Then
MbusQuery = Chr(0) + Chr(0) + Chr(0) + Chr(0) + Chr(0) + Chr(6) + Chr(1) + Chr(3)
+ Chr(0) + Chr(0) + Chr(0) + Chr(15)
Winsock1.SendData MbusQuery
ModbusWait = True
ModbusTimeOut = 0
Timer1.Enabled = True
Else
MsgBox ("Device not connected via TCP/IP")
Timer2.Enabled = False
End If
End Sub
```

Fuente: Autores.