DISEÑO Y CONSTRUCCIÓN DE UN BANCO DE EXPERIMENTACIÓN RELACIONADO CON LA IMPLEMENTACIÓN DE MICROCONTROLADORES Y CONTROLADORES LÓGICOS PROGRAMABLES EN SISTEMAS ELECTROHIDRÁULICOS

ALVARO DIAZ JIMENEZ
DAVID ERNESTO AGUILAR RODRÍGUEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍA MECÁNICA
BUCARAMANGA
2011

DISEÑO Y CONSTRUCCIÓN DE UN BANCO DE EXPERIMENTACIÓN RELACIONADO CON LA IMPLEMENTACIÓN DE MICROCONTROLADORES Y CONTROLADORES LÓGICOS PROGRAMABLES EN SISTEMAS ELECTROHIDRÁULICOS

ALVARO DIAZ JIMENEZ
DAVID ERNESTO AGUILAR RODRÍGUEZ

Trabajo de Grado para optar al título de Ingeniero Mecánico

Director
CARLOS BORRAS PINILLA
Ingeniero Mecánico

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍA MECÁNICA
BUCARAMANGA
2011

DEDICATORIA

A Dios por darme fuerzas en los momentos difíciles, a la memoria de mi padre, a mi hermana Edith y mi madre Olfa por su inmenso apoyo y comprensión.

Alvaro Diaz Jimenez

DEDICATORIA

A Dios por todas sus bendiciones, por esta hermosa familia y por ser mi mejor maestro.

A mi madre Luz María por ser ese ejemplo de honestidad fortaleza y dedicación, por su amor y respaldo incondicional.

A mi padre Hugo por darme la vida.

A mis hermanos Hugo y Magda por su amor, alegría, y su ejemplo de Unidad.

A mi hermosa esposa Ivonne y mis pequeñas hijas María Sofía y Mariana por ser mi punto de referencia y la luz de mis ojos.

David Ernesto Aguilar Rodríguez

AGRADECIMIENTOS

A Carlos Borras Pinilla, ingeniero mecánico, director del proyecto por su colaboración y respaldo.

A los profesores que en trayecto de la carrera brindaron sus conocimientos para fórmanos como profesionales.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

NOMENCLATURA O GLOSARIO

Entradas discretas. Una de las veinte conexiones de un dispositivo de campo que el PLC convierte a una señal eléctrica en un estado binario (OFF u ON), que es leído por la CPU en cada barrido del PLC.

Salidas discretas. Una de las dieciséis conexiones de salida del PLC que convierte un resultado interno del programa (ON u OFF) para activar un dispositivo de salida. Esto permite que el programa active grandes cargas.

Común de E/S. Una conexión en los terminales de entrada o de salida que es compartida por múltiple circuitos. Es generalmente en la trayectoria de retorno de la fuente de poder del circuito de E/S.

Rango de voltaje de entrada. El rango de voltajes de operación permitido en el circuito de entrada.

Corriente de entrada. Corriente de funcionamiento típica para una entrada activa (ENCENDIDA).

Indicadores de estado. Los LED que indican el estado de un punto de entrada o de salida. Todos los LED en el PLC DL06 están eléctricamente en el Lado Lógico del circuito de entrada o de salida.

Programa Ladder (o de escalera). El programa en el PLC con lógica booleana que simula un circuito eléctrico (Ladder es la palabra en inglés)

RESUMEN

TÍTULO:

DISEÑO Y CONSTRUCCIÓN DE UN BANCO DE EXPERIMENTACIÓN RELACIONADO CON LA IMPLEMENTACIÓN DE MICROCONTROLADORES Y CONTROLADORES LÓGICOS PROGRAMABLES EN SISTEMAS ELECTROHIDRÁULICOS*[*]

AUTORES:

ALVARO DIAZ JIMENEZ, DAVID ERNESTO AGUILAR RODRÍGUEZ [**]

PALABRAS CLAVES:

PLC, Microcontrolador, Automatización industrial, Autómatas programables, Arduino.

DESCRIPCIÓN:

La automatización industrial ha aportado de manera significativa en el crecimiento incesante de la industria, permitiendo que esta se adapte al panorama que exige un mundo globalizado, creando un ambiente más competitivo, ofreciendo nuevos y mejores servicios a sus clientes. Dentro del marco exigente que se genera a raíz de esta evolución, el ingeniero mecánico ha entrado a formar parte de este cambio, lo cual conduce a la necesidad de formar profesionales con conocimientos aptos en esta área, de esta manera contar con herramientas que le permitan afrontar los retos que la industria moderna le exige.

Por medio de este trabajo de grado se busca fomentar las capacidades de los estudiantes de ingeniería mecánica en el área de automatización industrial, puntualmente en el área de control en sistemas electrohidráulicos. Esto realizará mediante la construcción de un banco de experimentación, en el cual los estudiantes apreciaran y tendrán la oportunidad de realizar experiencias didácticas usando Controladores Lógicos Programables (PLC's) y Microcontroladores en el control de sistemas electrohidráulicos. Para ello se utilizará la unidad hidráulica existente del banco de prensas y malacate del laboratorio de potencia fluida de la Universidad Industrial de Santander.

El banco de experimentación cuenta con un soporte documental, el cual está constituido por un manual teórico, un manual de prácticas, y un manual de programación para el PLC y el Microcontrolador.

[*] Proyecto de Grado

[**] Facultad de ingenierías Físico-mecánicas, Escuela de Ingeniería Mecánica. Director CARLOS BORRAS PINILLA, Ingeniero Mecánico

SUMMARY

TITLE:

DESIGN AND CONSTRUCTION OF A EXPERIMENTATION BANK RELATING TO THE IMPLEMENTATION OF LOGIC CONTROLLERS PROGRAMMABLE MICROCONTROLLER AND SYSTEMS ELECTROHYDRAULIC[*]

AUTHORS:

Alvaro Diaz Jimenez.

David Ernesto Aguilar Rodríguez [**]

KEY WORDS:

PLC, Microcontroller, industrial automation, PLCs, Arduino.

DESCRIPTION:

Industrial automation has contributed significantly to the relentless growth of the industry, allowing it to fit the scenario that requires a globalized world, creating a more competitive environment, offering new and better services to their customers. Within the rigorous framework that is generated as a result of this evolution, the mechanical engineer has become part of this change, which leads to the need to train qualified professionals with expertise in this area, so having tools to the challenges that modern industry demands.

Through this thesis, we seek to promote the skills of mechanical engineering students in the area of industrial automation, occasionally in the area of electro-hydraulic control systems. This is done through the construction of a bank of experimentation, in which students appreciate and have the opportunity to make learning experiences using Programmable Logic Controllers (PLC's) and microcontrollers in the control of electro-hydraulic systems. This will use the existing hydraulic unit and winch bench presses fluid power laboratory of the Universidad Industrial de Santander.

The experimentation bank const on a documental support, which consists of a theoretical manual, a manual of practice, and a programming manual for the PLC and Microcontroller.

[*] Degree Work

[**] Faculty of Physical-Mechanical Engineering, Mechanical Engineering School, CARLOS BORRAS PINILLA Director, Mechanical Engineer

INTRODUCCIÓN

El avance tecnológico ha permitido que las empresas presenten una gran capacidad innovadora, potenciando su flexibilidad para adaptarse al nuevo panorama que nos entrega el mundo globalizado, ofreciendo nuevos servicios, favoreciendo a sus clientes en beneficios tangibles como el ahorro de Energía, la integración de procesos de alta complejidad, incorporando nuevos dispositivos de control automáticos que han contribuido significativamente tanto en la optimización como en la informatización de los procesos productivos, así como en la mejora en la gestión de la empresa.

Cincuenta años atrás se utilizaban los automatismos electromecánicos para realizar controles en diversos procesos, pero los problemas en la utilización de grandes espacios en los equipos, los elevados consumos de energía, las dificultades en las modificaciones, mantenimiento, detección y corrección de fallas, hicieron que este tipo de dispositivos estén actualmente en desuso.

En el laboratorio de la asignatura potencia fluida de ingeniería mecánica de la Universidad Industrial de Santander, los estudiantes realizan prácticas relacionadas con implementación de la lógica eléctrica cableada por medio de contactores de potencia, para el control electrohidráulico secuencial del banco de prensas y malacate, pero no cuentan con un medio físico que permita conocer, aprender, y realizar experiencias relacionadas con otras alternativas de control que se implementan actualmente en la industria, entre ellas, los Microcontroladores y los Controladores Lógicos Programables (PLC's).

Este trabajo de grado busca aportar en el proceso de enseñanza y aprendizaje de la asignatura Potencia fluida e ingeniería de control, mediante el diseño y construcción de un banco de experimentación, relacionado con la implementación de Microcontroladores y Controladores Lógicos Programables (PLC´s), en el control electrohidráulico secuencial de unidades de potencia hidráulica, utilizando

la estructura hidráulica del banco didáctico de prensa y malacate. Este banco de experimentación permitirá que los estudiantes actualicen los conocimientos relacionados con los sistemas de control que se implementan actualmente a nivel industrial. Esto se realizará de forma paralela al sistema de control actual, el cual se basa lógica eléctrica cableada por medio de contactores de potencia. Se utilizará la estructura hidráulica existente del banco de prensa y malacate del laboratorio de Potencia Fluida, brindando la posibilidad al estudiante de ingeniería mecánica de la universidad industrial de Santander de aprender y realizar prácticas relacionadas con esta temática.

La empresa FESTO ofrece a sus clientes bancos didácticos con las características requeridas para este proyecto, pero su elevado costo hace inviable esta alternativa. Adicional a esto, se realizó una evaluación de alternativas, teniendo en cuenta la ponderación obtenida por medio del despliegue de la función de calidad QFD, concluyendo de una manera cuantitativa, que la alternativa tres (Diseño y construcción un sistema de control secuencial basado en un Controlador Lógico programable y un Microcontrolador adaptado a la estructura hidráulica existente) es la que mejor se adapta a las necesidades de este proyecto.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Contribuir con la misión de la escuela de Ingeniería Mecánica de la Universidad Industrial de Santander en la formación académica de los estudiantes de pregrado, mediante el diseño y construcción de un banco didáctico para realizar experiencias relacionadas con la implementación de Microcontroladores y Controladores Lógicos Programables, en el control electrohidráulico secuencial de unidades de potencia hidráulica, permitiendo que los estudiantes actualicen los conocimientos acerca de estas alternativas de control programable, y adicionalmente proporcionar materiales de soporte para la enseñanza y aprendizaje de la asignatura de Potencia Fluida e ingeniería de control.

1.2 OBJETIVOS ESPECÍFICOS

- ✓ Diseñar y construir un banco de experimentación para la enseñanza y aprendizaje de la asignatura Potencia Fluida e Ingeniería de Control, relacionado con la implementación de los Microcontroladores y Controladores lógicos Programables, en el control electrohidráulico secuencial en unidades de potencia hidráulica, que sea adaptable al sistema de potencia hidráulica existente (banco didáctico de prensa y malacate) en el laboratorio de potencia fluida.

- ✓ Diseñar e instalar un sistema de control secuencial moderno, paralelo al existente (control lógico eléctrico cableado por medio de contactores de potencia), para realizar prácticas en el banco didáctico de prensa y malacate del laboratorio de la asignatura Potencia Fluida e Ingeniería de Control, usando un Controlador Lógico Programable DIREC KOYO DL06 y un Microcontrolador ATMEL-ATMEGA 1280.

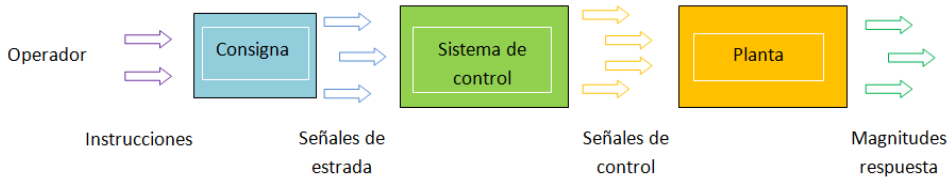
- ✓ Elaborar un manual para realizar prácticas relacionadas con la implementación de los Microcontroladores y Controladores Lógicos Programables, en el control electrohidráulico secuencial, del banco didáctico de prensa y malacate.

2. MARCO TEÓRICO – SISTEMA DE CONTROL

El concepto de control es extraordinariamente amplio, abarcando desde un simple interruptor que gobierna el encendido de una bombilla o el grifo que regula el paso de agua en una tubería, hasta el más complejo ordenador de proceso o el piloto automático de un avión.

Se puede definir el control como la manipulación indirecta de las magnitudes de un sistema denominado planta a través de otro sistema llamado sistema de control[1]. En la figura 1 se puede apreciar el esquema básico de un sistema de control.

Figura 1. Esquema básico de un sistema de control.



Fuente: Autores.

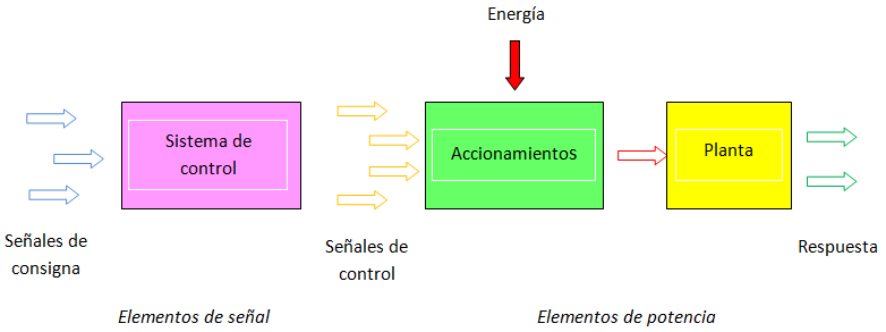
2.1 CLASIFICACIÓN DE LOS SISTEMAS DE CONTROL.

2.1.1 Sistemas de control de lazo abierto y de lazo cerrado. De acuerdo a la relación que existe entre la señal de entrada y la señal de salida, los sistemas de control se clasifican en sistemas de control de lazo abierto y sistemas de control de lazo cerrado.

[1] Fuente: Balcells, josep: Romeral, José Luis. “Autómatas programables”, Alfaomega Marcombo Boixareu Editores, 2000.

2.1.1.1 Sistemas de control de lazo abierto. Este tipo de sistema de control se caracteriza por que la señal de salida se basa en la señal de entrada pero es independiente de ella, es decir no existe retroalimentación. En la figura 2 se muestra una representación general de un sistema de control de lazo abierto.

Figura 2. Representación de un sistema de lazo abierto.



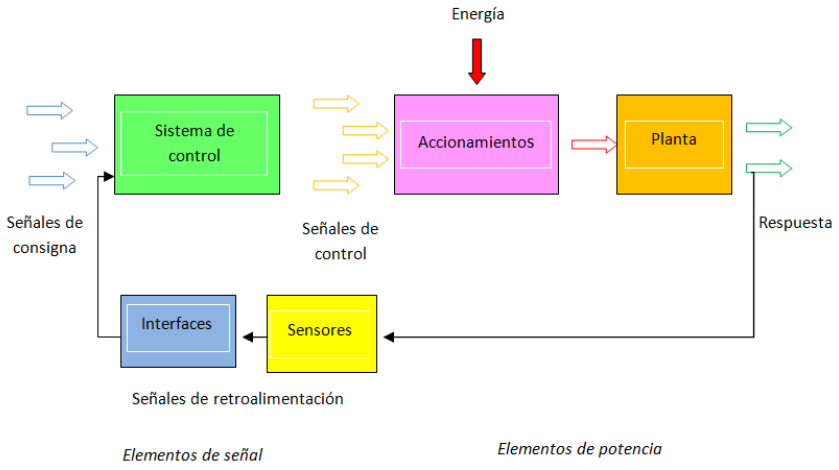
Fuente: Autores.

Elementos básicos. Los elementos básicos de un sistema de control de lazo abierto son los siguientes:

- Elemento de control: Determina la acción a tomar una vez se ha detectado una entrada en el sistema de control.
- Elemento de corrección (accionamientos): Responde a la entrada proveniente del elemento de control iniciando una acción que genera un cambio en la variable controlada al valor que se desea.
- Proceso o planta: Es el proceso o planta del sistema en el que se quiere controlar una variable.

2.1.1.2 Sistemas de control de lazo cerrado. En este tipo de sistema de control, la señal de salida tiene un efecto directo sobre la señal de entrada, en este caso existe retroalimentación, esto se puede observar en la figura 3.

Figura 3. Representación de un sistema de lazo cerrado.



Fuente: Autores.

La diferencia entre la señal de entrada y la señal de retroalimentación crean una señal de error actuante la cual entra a un detector ubicado dentro del sistema de control, para posteriormente corregir la señal de salida y adaptarla al valor deseado.

Elementos básicos. Los elementos básicos de un sistema de control de lazo abierto son los siguientes:

- Elemento de comparación: Este compara el valor que se requiere con el valor que medido que se obtiene en la salida, luego genera una señal de error que indica la diferencia entre el valor leído en la salida y el valor que se requiere para obtener la salida deseada.
- Elemento de control: Decide la acción que se debe tomar una vez recibida la señal de error.
- Elemento de corrección (accionamientos): Se encarga de eliminar el error y producir un cambio en el proceso.
- Elemento de proceso: Es el proceso o planta del sistema en el que se quiere controlar una variable.

- Elemento de medición (sensor). se encarga de proporcionar la señal de retroalimentación y enviarla al elemento de comparación para que este determine si existe o no un error.

2.1.2 Sistemas De Control Analógicos Y Digitales[2]. Según la naturaleza de las señales que intervienen en el proceso, los sistemas de control pueden dividirse en los siguientes grupos:

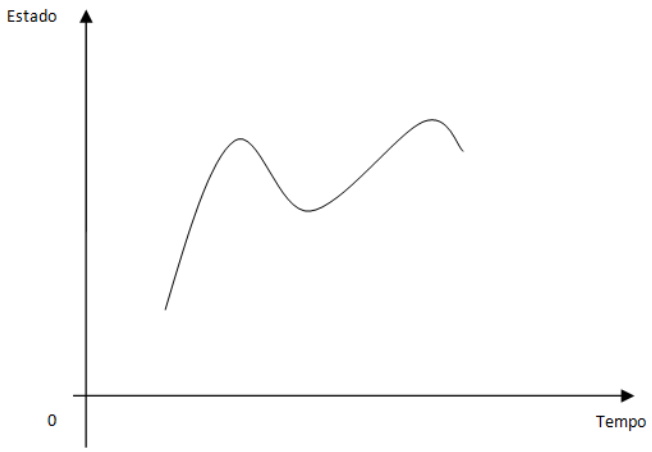
- Sistemas analógicos.
- Sistemas digitales.
- Sistemas híbridos analógico-digitales.

2.1.2.1 Sistemas analógicos. Trabajan con señales de tipo continuo, con un margen de variación determinado. Un ejemplo de este tipo de señales se puede observar en la figura 4. Dichas señales suelen representar magnitudes físicas del proceso, mediante una tensión o corriente proporcionales a su valor (0 a 10 V, 4 a 20 mA, etc.). Ejemplo de este tipo de señales son los siguientes:

- Temperatura.
- Velocidad.
- Presión.
- Flujo nivel.

[2] Fuente: Balcells, josep: Romeral, José Luis. “Autómatas programables”, Alfaomega Marcombo Boixareu Editores, 2000. p. 24.

Figura 4. Esquema señal analógica.



Fuente: Autores.

2.1.2.2 Sistemas digitales. Trabajan con señales todo o nada, llamadas también binarias, que sólo pueden presentar dos estados o niveles: abierto o cerrado, conduce o no conduce. Estos niveles o estados se suelen representar por variables lógicas o bits, cuyo valor puede ser sólo 1 o 0, empleando la notación binaria del álgebra de Boole, esto se puede observar en la figura 5. Ejemplo de este tipo de señales son los siguientes:

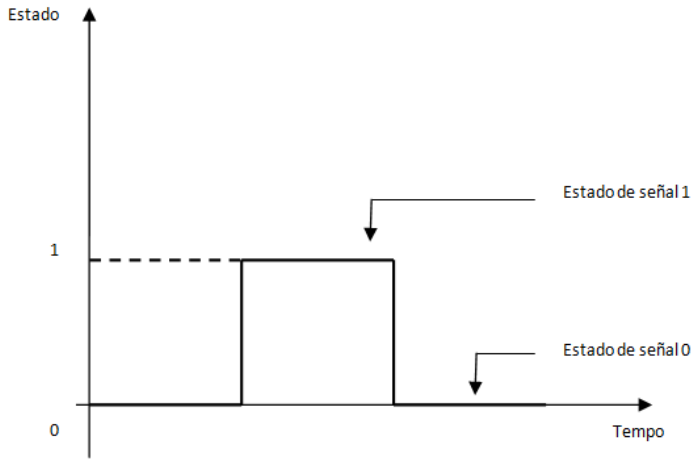
Entrada:

- Pulsador.
- Interruptor deposición.
- Interruptor fotoeléctrico.

Salida:

- Contactor.
- Lámpara indicadora.

Figura 5. Esquema señal digital.



Fuente: Autores.

2.1.2.3 Sistemas híbridos. De control actuales con un cierto grado de complejidad, y en particular los autómatas programables, son casi siempre, es decir, sistemas que procesan a la vez señales analógicas y digitales. No obstante, se tiende a que la unidad de control sea totalmente digital y basada en un microprocesador, que aporta la capacidad de cálculo necesaria para tratar las señales todo o nada en forma de bits y las señales analógicas numéricamente.

2.1.3 Sistemas de control cableados y programables. De acuerdo a la configuración y a su construcción estos se pueden clasificar de la siguiente manera:

- Sistemas cableados (poco adaptables).
- Sistemas programables (muy adaptables.

En la tabla 1 se pueden observar las principales características de los sistemas cableados y los sistemas programables y se realiza una comparación entre ellas.

Tabla 1. Comparación entre sistemas cableados y sistemas programables.

CARACTERISTICA	CABLEADO	PROGRAMABLES
Flexibilidad de adaptación al proceso	Baja	Alta
Hardware estándar para dist. Aplicaciones	No	Si
Posibilidades de ampliación	Bajas	Alta
Interconexiones y cableado exterior	Mucho	Poco
Tiempo de desarrollo del proyecto	Largo	Corto
Posibilidades de modificación	Difícil	Fácil
Mantenimiento	Difícil	Fácil
Herramientas para prueba	No	Si
Stocks de mantenimiento	Medios	Bajos
Modificaciones sin parar el proceso	No	Si
Coste para pequeñas series	Alto	Bajos
Estructuración en bloques independientes	Difícil	Fácil

Fuente: Autores.

2.1.3.1 Sistemas de control cableado. Estos realizan una función de control fija, la cual depende directamente de los elementos que lo constituyen así como también de la forma como se encuentren interconectados. Si se desea modificar su función de control, se deben cambiar sus componentes o modificar sus conexiones.

2.1.3.2 Sistemas de control programables. Estos pueden realizar distintas funciones de control sin necesidad de alterar su configuración física, ya que por medio de un programa de control se pueden variar estas funciones. Debido a lo expuesto anteriormente los sistemas de control programables han venido sustituyendo a los sistemas de control cableado.

2.2 AUTÓMATA PROGRAMABLE.

Es un equipo electrónico de control con cableado interno independiente al proceso a controlar, este equipo se adapta a dicho proceso por medio de un programa específico, en el cual se programan las secuencias de operaciones a realizar en el proceso que se desea controlar. Las secuencias de operaciones se definen con relación en las señales de entrada y salida del proceso, cableadas directamente en los bornes de conexión del autómata programable.

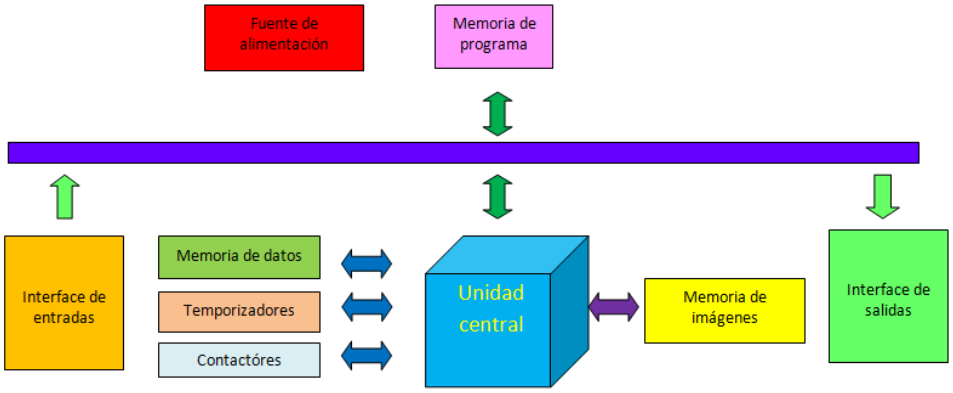
Las señales de entrada provienen de elementos analógicos como dispositivos de salida en tensión o corriente continúa, sensores de temperaturas, o de dispositivos digitales como detectores de proximidad, límites de carrera. Las señales de salida pueden ser de tipo analógicas en tensión o corriente, o de tipo digital todo o nada, estas señales de salidas se envían a los preactuadores o actuadores como motores, electroválvulas, contactores, entre otros.

2.2.1 Constitución del autómata programable. Estos equipos se constituyen básicamente en los siguientes componentes:

- Memorias.
- Unidad central de proceso o de control (CPU).
- Interfaces de entra y salida.
- Fuente de alimentación.

La figura 6 muestra por medio de un diagrama de bloques, el esquema básico de un autómata programable.

Figura 6. Diagrama de bloques de un autómata programable.



Fuente: Autores.

2.2.1.1 Memorias del autómata. Guarda todos los datos necesarios para realizar la tarea de control, estos datos son:

Datos del proceso:

- Señales de planta, entradas y salidas.
- Datos alfanuméricos y constantes.
- Variables internas, bit y de palabra.

Datos de control:

- Configuración del autómata (modo de funcionamiento, número de entradas y salidas conectadas, etc.).
- Instrucciones del usuario (programa).

De acuerdo a las características de lectura y escritura las memorias de un autómata se clasifican en:

Memorias ROM. De solo lectura no reprogramables ya pueden ser modificadas de ninguna manera, y se utilizan para almacenar el programa monitor que contiene tareas como iniciación tras puesta en tensión o reset, intercambio de información con unidades exteriores, lectura y escritura en las interfaces de E/S.

Memorias RAM. Pueden ser leídas y modificadas cuantas veces sea necesario por medio de los buses internos, se utilizan como memorias de datos internos, y únicamente como memorias de programa en caso que pueda asegurarse el mantenimiento de datos con una batería externa. Tiene poca capacidad de almacenamiento.

Memorias EPROM. Son de solo lectura y una vez borrado su contenido pueden programarse con un circuito especial, son utilizadas para almacenar el programa del usuario.

Memorias EEPROM. Son reprogramables y puede ser alterada por medios eléctricos, es decir, reprogramables sobre el propio circuito, sin necesidad de extracción y borrado exterior, se utilizan para almacenar programas.

Memoria interna**.** Almacena el estado de las variables que maneja el autómata, como entradas, salidas, relés internos, señales de estado, contactores, etc. La clasificación de estas memorias no se realiza atendiendo sus características de lectura y escritura, si no por el tipo de variables que almacena y el número de bits.

Posiciones de 1 bits:

- Relés internos.
- Relés especiales/auxiliares.
- Memoria de imagen de entrada/salidas.

Posiciones de 8, 16 o más bits:

- Temporizadores.
- Contactores.

Memoria de programa**.** Almacena el programa escrito por el usuario para su aplicación, en ella se guarda la secuencia de operaciones que deben realizarse con relación a las señales de entrada para obtener señales de salida, también contiene los parámetros de configuración del autómata.

2.2.1.2 Unidad central de proceso o de control (CPU).[3] Se encarga de ejecutar el programa que el usuario ha ingresado, verificando el estado de las entradas y tomando de la memoria de programa las secuencias de instrucciones que se deben ejecutar, y elabora a partir de estas, las señales de salida.

También puede establecer comunicación con periféricos externos como, monitores LED/LCD, otros ordenadores o autómatas, unidad de programación, etc. La CPU está constituida principalmente por las siguientes partes:

ALU (Aricmetic Logic Unit). Es la encargada de realizar las operaciones aritméticas y lógicas.

Acumulador. Almacena el resultado de la última operación que ha realizado el ALU.

Flags. Indica el resultado de la operación (mayor que, positivo, negativo, resultado uno o cero).

[3] Fuente: Balcells, josep: Romeral, José Luis. “Autómatas programables”, Alfaomega Marcombo Boixareu Editores, 2000. p. 20.

Contador de programa. Se encarga de la lectura de las instrucciones del usuario, y de la secuencia de la ejecución.

Decodificador de instrucciones y secuenciador. Es el cableado y programa en donde se decodifican las instrucciones que se han leído en la memoria, y luego genera las señales de control.

Programa ROM monitor del sistema. En este programa se almacenan las secuencias de puesta en marcha, como lo son las rutinas de tests, y de error en ejecución, etc.

2.2.1.3 Interface de entradas y salidas. Se encargan de establecer la comunicación entre el autómata y la planta, las señales se conectan a través del proceso por medio de unas borneras y con el autómata por medio de unas bus interno, además filtran, adaptan y codifican de forma comprensible estas señales antes de enviarlas a la unidad central de proceso. Debido a la gran cantidad de variantes que se pueden presentar en las señales las interfaces se pueden clasificar de la siguiente manera:

De acuerdo a la señal de alimentación:

- De corriente continua (estáticas de 24/110Vcc).
- De corriente continua a colector (NPN, PNP).
- De corriente alterna (60/110/220Vca.
- Salidas por relé (libres de tensión).

Por aislamiento:

- Con acoplamiento directo.
- Con separación galvánica (optoacopladores).

Por la forma de comunicación de la unidad central:

- Comunicación en serie.
- Comunicación en paralelo.

Por la ubicación:

- Locales.
- Remotos.

2.2.1.4 Fuente de alimentación. Por medio de una tensión exterior, proporciona las tensiones necesarias para el óptimo funcionamiento de los distintos circuitos electrónicos del sistema. Un autómata programable está constituido por componentes que requieren diferentes tipos de tensiones y potencias. Estos componentes también estas sometidos a condicione de ruidos electromagnéticos diferentes. Por ello es frecuente que la alimentación se realice procurando independizar las siguientes partes del circuito:

- Alimentación entradas.
- Alimentación salidas.
- Unidad central de interfaces.

2.2.2 Controlador Lógico Programable (PLC). Es un autómata programable diseñado para controlar un proceso o una máquina, tiene la capacidad de ser programado o reprogramado rápidamente según la necesidad de la aplicación. Este dispositivo fue inventado para remplazar los circuitos secuenciales basados en relés y temporizadores que se utilizaban en el control de las máquinas. El PLC funciona monitoreando sus entradas, y dependiendo de su estado, activando y desactivando sus salidas. El operador introduce al PLC un programa, usualmente vía Software, lo que permite cambiar los parámetros de control. Los PLCs son usados en muchas aplicaciones:

- Reemplazar la lógica de relés para el comando de motores, máquinas, cilindros, neumáticos e hidráulicos, etc.
- Regulación de aparatos remotos desde un punto de la fábrica.
- Efectuar diagnósticos de fallas y alarmas.
- Controlar y comandar tareas repetitivas y peligrosas.
- Reemplazar temporizadores y contadores electromecánicos.

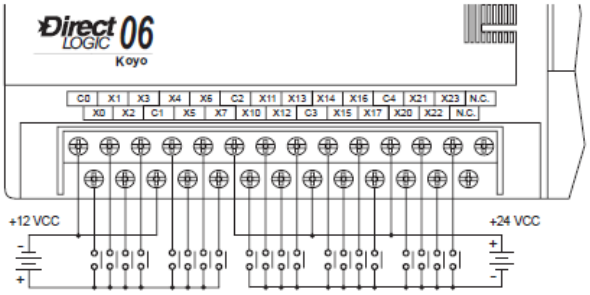
Sus principales beneficios son:

- Mayor facilidad para el mantenimiento y puesta en servicio.
- Menor cableado, reduce los costos y los tiempos de parada de planta.
- Reducción del espacio en los tableros.

2.2.2.1 Principales características de los PLCs.[4]

Capacidad modular de entradas / salidas. Esto permite la combinación de distintos niveles y tipos de señal de entrada, así como también el manejo de salidas para distintos tipos de carga, como se puede observar en la figura 7. Igualmente si la aplicación crece, y se requiere mayor número de entradas/salidas, casi sin ningún problema los PLCs pueden adecuarse al nuevo requerimiento.

Figura 7. Capacidad modular entradas/salidas.

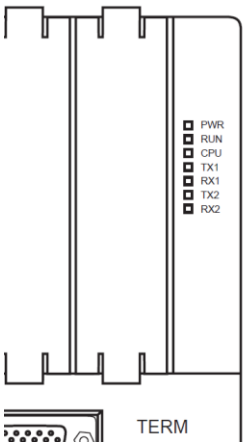


Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 60.

[4] Danilo Navarro; Controlador Lógico Programable (PLC). Universidad De Oriente. Julio 2001.

Autodiagnóstico de fallas. El PLC monitorea el funcionamiento de su CPU, Memoria y circuito de interfaces de entradas y salidas, e igualmente monitorea el correcto funcionamiento del programa de aplicación. En ambos casos señaliza por medio de LEDs en su cara frontal el estado respectivo. Obviamente esta capacidad es de gran utilidad para efectos de mantenimiento y corrección de fallas. Estos LEDs se pueden observar en la figura 8.

Figura 8. Leds de estado.



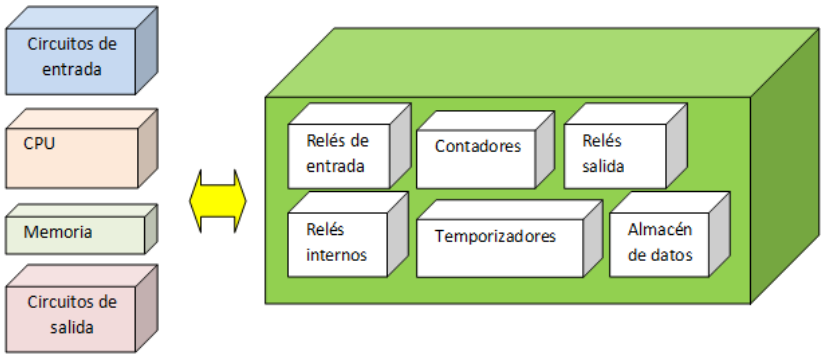
Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición. 2009. p. 34.

Programación de la lógica de control. Esto permite la fácil adaptación a los cambios en la lógica de operación de las máquinas y procesos.

Capacidad para generar reportes y comunicarse con otros sistemas. Con esta facilidad se pueden integrar interfaces de explotación Hombre-Máquina, sacándole al sistema mayor cantidad de información. Igualmente los PLCs pueden participar en redes de datos comunicándose con otros PLCs para formar sistemas de control distribuidos, o integrándose a las redes administrativas de la producción.

2.2.2.2 Estructura interna. Los PLCs están constituidos principalmente de una CPU, área de memorias, y circuitos apropiados de entradas y salidas para datos. El PLC tiene la capacidad de simular por medio del software cientos de Relés, temporizadores, contadores, almacén de datos. En la figura 9 se puede apreciar la estructura interna de un PLC.

Figura 9. Estructura interna del PLC.



Fuente: Autores.

Relés de entrada. Por lo general son transistores que funcionan como Relés estáticos, están conectados a las entradas del PLC y reciben señal del exterior proveniente de, interruptores de final de carrera, sensores, etc.

Relés internos. No se conectan al mundo exterior, y en realidad no existen físicamente, estos son simulados por medio del software del equipo, esto hace que los PLCs sean de menor tamaño con posibilidades casi infinitas.

Contadores. No existen físicamente y también son simulados mediante software, por lo general cuentan de forma ascendente y descendente.

Temporizadores. Tampoco existen físicamente. Y son simulados de igual manera que los contadores, son de varios tipos al reposo y al trabajo[5].

Relés de salida. Se conectan a los dispositivos de campo, y existen físicamente, por lo general están constituidos por transistores, Relés electromecánicos o Triacs, dependiendo del modelo escogido.

Almacenamiento de datos. Se utilizan para el almacenamiento temporal de para manipulaciones matemáticas o de datos, se utilizan también para almacenar datos cuando se corta el suministro de energía.

2.2.2.3 Funcionamiento[6]. Un PLC trabaja realizando continuamente un barrido (SCAN) sobre un programa. Este ciclo de barrido o scan consta principalmente de 4 pasos:

Diagnóstico interno. En este paso el PLC revisa su circuitería interna en busca de defectos de entradas, salidas, CPU, memorias y batería. También revisa el WATCHDOG[7] y los desbordamientos de memoria para revisar fallas en el programa de aplicación.

Chequear el estado de las entradas. Al principio el PLC accede cada una de las entradas para determinar si están activadas o desactivadas (on/off). Es decir, ¿Está activado el sensor conectado a la primera entrada?, ¿El segundo?, ¿El tercero? … Luego el PLC graba estos datos en la tabla imagen de proceso para usarlos en el próximo paso.

---

[5] Se denominan Relés de trabajo aquellos que se cierran cuando su bobina es alimentada y Relés de reposo a los cerrados en ausencia de alimentación de la misma.

[6] Danilo Navarro; Controlador Lógico Programable (PLC). Universidad De Oriente. Julio 2001.

[7] WATCHDOG es un concepto de protección usado para volver a reiniciar el programa cuando éste "se pierde" o realiza una acción no prevista.

Ejecutar el programa de la aplicación. El PLC ejecuta el programa de la aplicación creada por el usuario, una instrucción a la vez. Por ejemplo, si el programa especifica que si la primera entrada esta “on” se debe activar la salida numero 2, el PLC graba este resultado para tenerlo en cuenta en el próximo paso. Como ya el PLC conoce cuales entradas están activadas o desactivadas (paso 2), él será capaz de decidir cuales salidas se deben activar basado en el estado de las entradas y en el estado de los contadores, temporizadores y bits internos. Como ya se dijo el PLC guarda este resultado para usarlo en el próximo paso.

Actualizar el estado de las salidas. Finalmente el PLC actualiza el estado de las salidas basado en los resultados lógicos del paso 3. Siguiendo el ejemplo del paso 3, el PLC activará en este tercer paso la salida numero 2 basado en el hecho que la primera entrada estaba en “on”. Después del cuarto paso el PLC vuelve al paso uno y repite la rutina continuamente. Así, un SCAN se define como el tiempo que toma el PLC para ejecutar los cuatro pasos descritos anteriormente.

2.2.2.4 Conexión de entradas y salidas. Las tensiones de entradas y salida de estos dispositivos más comunes son DC (drenadoras o surtidoras de corriente) y AC. Los rangos típicos de tensión son los siguientes:

- 12 – 24 VDC.
- 100-120 VAC.
- 5 VDC (TTL).
- 200-240 VAC.
- 48 VDC.
- 24 VAC.

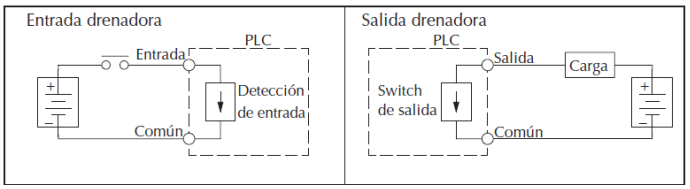
Realizando una comparación entre las entradas de tipo DC y AC se observan las siguientes características:

- Los voltajes de tipo DC pueden ser conectados a una gran variedad de equipos y sistemas eléctricos.
- Los voltajes DC usualmente son más bajos (12 – 24 V) y por lo tanto resulta menos riesgoso operar con ellos.
- Las entradas DC son bastantes rápidas. Las entradas AC requieren de mayor tiempo para ser reconocidas.
- Las señales AC son más inmunes al ruido que las señales DC, por eso pueden cubrir mayor distancia y ambientes ruidosos.
- El suministro AC es más fácil y menos costoso al momento de alimentar equipos eléctricos.
- Las señales AC son muy comunes en muchos equipos de automatización.

Entradas y salidas DC. Este tipo de entradas puede dividirse en dos tipos drenadoras y surtidoras:

Entradas y salidas de tipo drenadoras (NPN). Brindan una ruta para suministrar negativo. Para esta configuración se conecta el positivo a la entrada/salida del PLC mientras que el negativo se conecta al común, esto se pude apreciar en la figura 10.

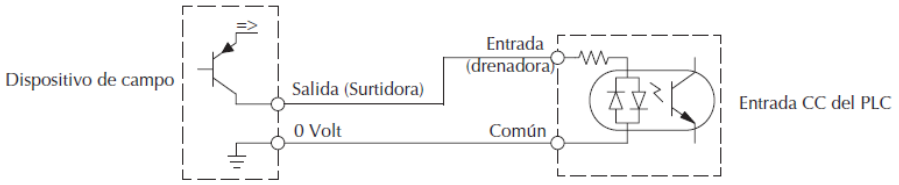
Figura 10. Entrada y salida tipo NPN.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 60.

Para conectar dispositivos como sensores a este tipo de entradas, estos deben tener una configuración opuesta (PNP) para que tenga compatibilidad con las entradas NPN del PLC, esto se puede observar en la figura 11.

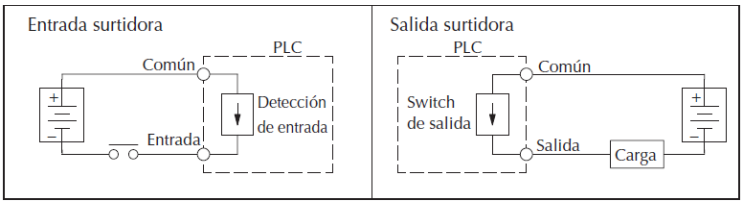
Figura 11. Conexión de dispositivo de campo a entradas tipo NPN.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 61.

Entradas y salidas de tipo surtidora (PNP). Brindan una ruta para suministrar positivo. Para esta configuración se conecta el negativo a la entrada/salida del PLC mientras que el positivo se conecta al común. Esto se pude observar en la figura 12.

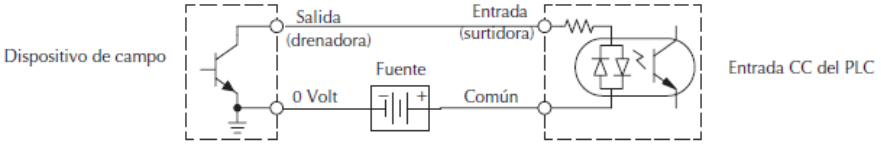
Figura 12. Entrada y salida PNP.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 61

Para conectar dispositivos como sensores a este tipo de entradas, estos deben tener una configuración opuesta (NPN) para que tenga compatibilidad con las entradas PNP del PLC, tal como se observa en la figura 13.

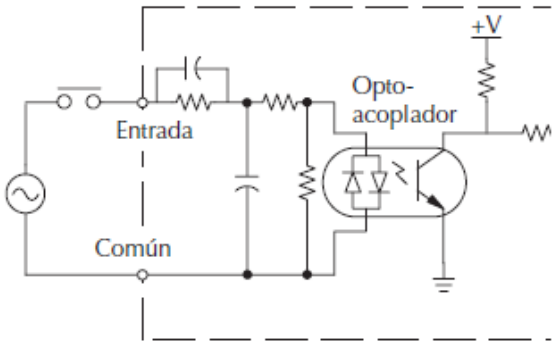
Figura 13. Conexión de dispositivo de campo a entradas tipo PNP.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 61.

Entradas y salidas AC. No posee polaridad es decir no hay positivo o negativo por el cual preocuparse, la desventaja de este tipo de tensiones es su peligrosidad si no se toman la precauciones debidas ya que manejan altas potencias. Los niveles más comunes son de 24, 28, 110, 220 Voltios. Los modulaos de entras de AC son poco utilizados debido a que gran parte de los dispositivos de campo como sensores entre otros, trabajan con tensiones a DC. La figura 14 muestra un ejemplo de conexión de entrada tipo AC.

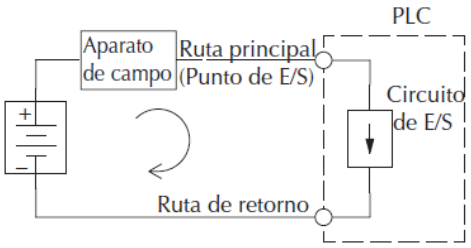
Figura 14. Conexión entrada tipo AC.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p.62.

Terminales comunes. Para que opere un circuito de entradas y salidas del PLC, la corriente debe entrar en un terminal y salir en otro. Esto significa que por lo menos dos terminales se asocian con cada punto de entrada o salida. En la figura 15 el terminal de entrada o salida es el camino principal para la corriente. Un terminal adicional debe proporcionar el camino de regreso a la alimentación. Este es el terminal común.

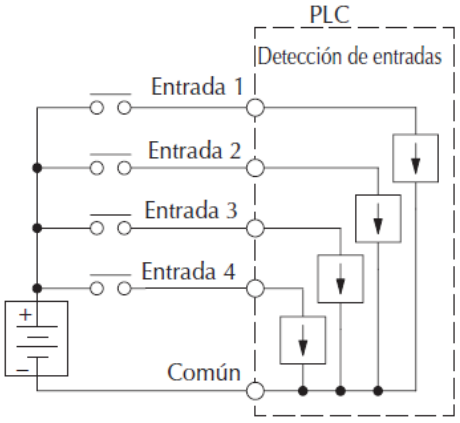
Figura 15. Ruta de retorno de la corriente.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 62.

La figura 16 muestra un grupo de 4 puntos de entradas que comparten un camino común de regreso, de esta manera, las cuatro entradas requieren sólo cinco terminales en vez de ocho.

Figura 16. Terminal común.



Fuente: Automatión Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición. 2009. p.63.

2.2.2.5 Programación. Básicamente existen dos tipos de programas en un PLC, que difieren de acuerdo a la accesibilidad por parte de los usuarios hacia ellos y de las funciones que realizan. Estos son:

- Programas del sistema.
- Programas de usuario.

Programas del sistema. Son los llamados Software y se encargan de las funciones operativas del PLC. Funciones como manejo de los dispositivos de entrada y salida, almacenamiento de la información durante largos periodos de tiempo, organizar el procesamiento de los programas de usuario, entre otras, las realiza el programa del sistema.

Programas de usuario. Son las instrucciones o proposiciones que el usuario programa, con el objetivo de resolver tareas de automatización específicas. Para ello el usuario escribe el programa en un lenguaje de programación que mejor se adapte a su trabajo y que le pueda brindar el fabricante del equipo.

La norma IEC 1131-3 define cinco lenguajes de programación que pueden ser usados para definir los procedimientos de control y automatización.

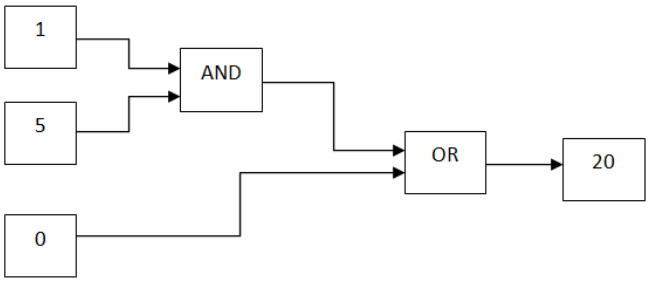
Gráficos secuenciales o GRAFCET. En este lenguaje se divide el ciclo de proceso etapas y transiciones, que se asocian respectivamente a acciones y condiciones. Las etapas representan las acciones a realizar, mientras que las transiciones representan las condiciones. La figura 17 muestra un ejemplo de programación tipo Grafcet.

Figura 17. Programación Diagrama Grafcet.

Fuente: Autores.

Diagrama de bloques funcionales. Permite que el usuario construya procedimientos mediante la unión de bloques funcionales que utilizan las puertas lógicas AND, OR, etc. La figura 18 muestra un ejemplo de programación tipo diagrama de bloques funcionales

Figura 18. Programación Diagrama de bloques funcionales.



Fuente: Autores.

Texto estructurado. Es de tipo Booleano de alto nivel y estructurado, incluye típicas sentencias como IF, THEN, ELSE, FOR, entre otras. Tienen gran aplicación cuando se requieren cálculos matemáticos, comparaciones, emular protocolos, etc. La figura 19 muestra un ejemplo de programación tipo texto estructurado.

Figura 19. Programación Texto estructurado.

```
000   LD     %10.2   Bp inicio ciclo
      AND    %10.3   DR entrando
004   AND    %10.4   HU presencia de vehículo
      AND    %10.8   FT cerca al pórtico
      OR     %10.1   YU presencia de peatón
```

Fuente: Autores.

Nemónicos o lista de instrucciones. Se basa en operaciones Booleanas, utilizan letras y números para su representación, posee una apariencia parecida al texto estructurado, pero su estructura varía de acuerdo con el fabricante. La figura 20 muestra un ejemplo de programación tipo nemónico.

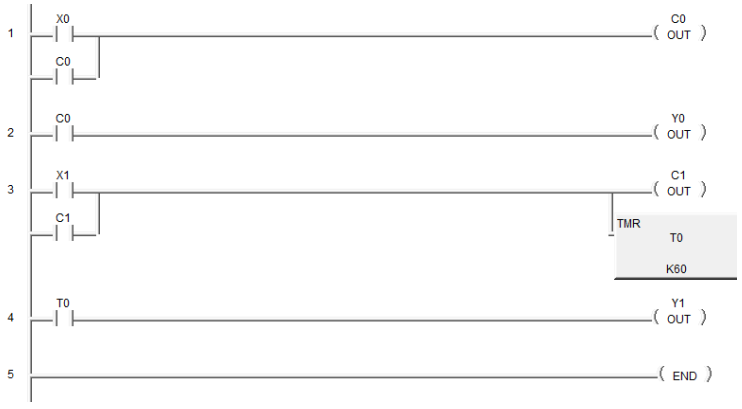
Figura 20. Programación Nemónica o lista de instrucciones.

Siemens	Telemecanique	General electric
U A0.1	L I0. 01	AND %I0001
U B0.2	O I0. 01	AND %I0002
U A0.3	A I0. 01	OR %I0003

Fuente: Autores.

Programación LADDER o Escalera. Tiene Cierta analogía con los sistemas de contactos según la norma Nema. Su estructura es semejante a la que se utiliza para representar los circuitos de control en la lógica cableada por Relé, contactores. La figura 21 muestra un ejemplo de programación tipo LADDER.

Figura 21. Programación LADDER o Escalera.



Fuente: Autores.

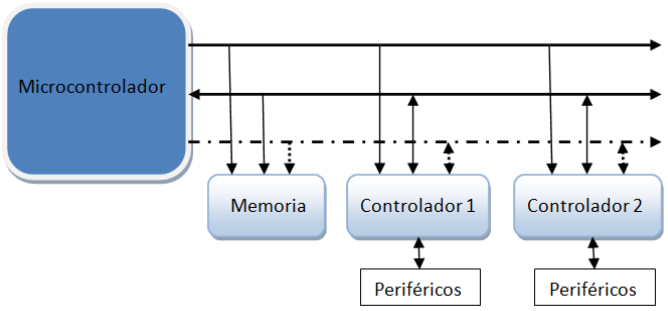
2.3 MICROCONTROLADOR.

Es un circuito integrado que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de «controlador incrustado» *(embedded controlle*r). Se dice que es “la solución en un chip” porque su reducido tamaño minimiza el número de componentes y el costo.

El Microcontrolador es un computador dedicado. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan la conexión de sensores y actuadores del dispositivo a controlar. Una vez programado y configurado el Microcontrolador solamente sirve para gobernar la tarea asignada.

2.3.1 Arquitectura. Aunque inicialmente todos los Microcontroladores adoptaron la arquitectura clásica de Von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de Von Neumann en la figura 22 se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control).

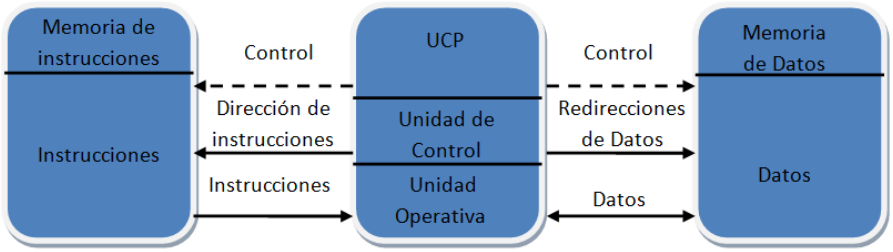
Figura 22. Arquitectura Von Neumann.



Fuente: Autores.

La arquitectura Harvard en la figura 23 dispone de dos memorias independientes una, que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.

Figura 23. Arquitectura Harvard de Microcontrolador.



Fuente: Autores

2.3.2 Recursos comunes de los Microcontroladores. Un microprocesador dispone normalmente de los siguientes componentes:

Procesador o ucp (unidad central de proceso). Es el elemento más importante del Microcontrolador y determina sus principales características, tanto a nivel de hardware como software.

Se encarga redireccionar la memoria de instrucciones, recibir el código OP de la instrucción de curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.

Memoria. En los Microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos.

Puertos de entrada y salida. La principal utilidad de las patillas que posee la cápsula que contiene un Microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores y según los controladores de periféricos que posea cada modelo de Microcontrolador, se destinan a proporcionar el soporte a las señales de entrada, salida y control.

Todos los Microcontroladores destinan algunas de sus patillas a soportar líneas de E/S de tipo digital, esto es, todo o nada. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos. Las líneas digitales de los puertos pueden configurarse como entrada o como salida.

Generalmente, el circuito de reloj está incorporado en el Microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo junto a elementos pasivos o bien un resonador cerámico o una red R-C. Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía y de calor generado.

2.3.3 Recursos especiales de los Microcontroladores. Los principales recursos específicos que incorporan los Microcontroladores son:

Temporizadores o "Timers". Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patillas del Microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

Perro guardián o "Watchdog". Cuando un ordenador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema.

En la mayoría de los casos y a diferencia de un ordenador personal, un Microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día y 365 días al año. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, el programa no refrescará al Perro guardián y, al completar su temporización, provocará el reset del sistema.

Protección ante fallo de alimentación o "Brownout". Se trata de un circuito que resetea al Microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo ("brownout"). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor. Esto es muy útil para evitar datos erróneos por transiciones y ruidos en la línea de alimentación.

Estado de reposo ó de bajo consumo. Son abundantes las situaciones reales de trabajo en que el Microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), los Microcontroladores disponen de una instrucción especial, que les pasa al estado

de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando sumido en un profundo "sueño" el Microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el Microcontrolador se despierta y reanuda su trabajo.

Conversor A/D (CAD). Los Microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patillas del circuito integrado.

Conversor D/A (CDA). Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patillas de la cápsula. Existen muchos dispositivos de salida que trabajan con señales analógicas.

Comparador analógico. Algunos modelos de Microcontroladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por una de las patillas de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.
También hay modelos de Microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

Modulador de anchura de impulsos o PWM. Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patillas del encapsulado. Resulta útil para sistemas de control de potencia, como por ejemplo motores.

Puertos de comunicación. Con objeto de dotar al Microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- UART, adaptador de comunicación serie asíncrona.
- USART, adaptador de comunicación serie síncrona y asíncrona
- Puerto paralelo esclavo para poder conectarse con los buses de otros microprocesadores.
- USB (Universal Serial Bus), el conocido bus serie para los PC.
- Bus I^2C, que es un interfaz serie de dos hilos desarrollado por Philips.
- Interface SPI, un puerto serie síncrono.
- CAN (Controller Área Network), para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles. En EE.UU. se usa el J185O.
- TCP/IP, ya existen Microcontroladores con un adaptador de comunicación para este protocolo.

2.3.4 Lenguajes de programación. Se han desarrollado muchos tipos de lenguaje para los Microcontroladores pero los más usados son:

Ensamblador. Es el lenguaje de bajo nivel natural de la línea del Microcontrolador tanto para gama baja, media o alta (Ver figura 24). Se pueden crear macros con este lenguaje, para después simplificar el código en diferentes desarrollos.

Figura 24. Ejemplo de lenguaje de programación Ensamblador.

```
    org 0x0000            ;org coloca el programa en el origen.
    movlw 0x07
    movwf CMCON           ;Deshabilito los comparadores

    bsf STATUS, RP0       ;entra a Bank 1 para poder acceder a TRISB
    clrf TRISB            ;Configuro PortB como salida
    bcf STATUS, RP0       ;retorna a de nuevo el Bank 0

Loop
    movlw b'00000001'     ;enciendo LED
    movwf PORTB
    nop                   ;nop no hace nada
    nop
    call Delay            ;Llamo a la rutina de retardo de 200ms
    movlw b'00000000'     ;apago LED
    movwf PORTB
    call Delay            ;Llamo la rutina de retardo de 200ms
    goto Loop             ;Inicia de nuevo el ciclo.

;****SUBRUTINA DE RETARDO****;
; Esta rutina retarda 200ms. ;
;****************************;
Delay
    movlw d'200'          ;Espero 200 ms (Para un reloj de 4 MHz)
    movwf count1
d1  movlw 0xC7
    movwf counta
    movlw 0x01
    movwf countb
Delay_0
    decfsz counta, f
    goto $+2
    decfsz countb, f
    goto Delay_0

    decfsz count1 ,f
    goto d1
    retlw 0x00

    end
```

Fuente: http://electronicadesarrollo.blogspot.com/2007/12/programacin-de-pics.html

Lenguaje C. Es un lenguaje de alto nivel más cercano a la máquina, con el se pueden construir rutinas matemáticas fácilmente (ver la Figura 25). Los programas al compilarlos pueden resultar un poco extensos y pesados por ello debe tenerse en cuenta la capacidad de memoria de programa del Microcontrolador a utilizar.

Figura 25. Ejemplo de lenguaje de programación en C.

```
//* Nombre del archivo: blink.c
//* Ejemplo de programa que hace parpadear un LED
//* conectado al PORTB.0
//* El pic utilizado es el 16F628A de la gama media.
//*********************************************************************

#include <16f628A.h>                  // Define el PIC
#fuses XT,NOWDT,NOPROTECT,PUT,BROWNOUT // configuración de fusibles

#use delay(clock=4000000)             // Cristal: 4 Mhz

#byte port_b=6                        // Dirección en RAM de PORTB

void main() {
   set_tris_b(0);                     // Configura Port B como salida
   port_b=0;

   while(TRUE) {                      // Bucle infinito
      port_b=0;                       // Apaga todo el puerto B
      delay_ms(200);                  // Espera 200ms
      port_b=1;                       // Enciende el bit 0.
      delay_ms(200);                  // Espera otros 200ms
   }
```

Fuente: http://electronicadesarrollo.blogspot.com/2007/12/programacin-de-pics.html

El lenguaje BASIC. Es un lenguaje muy simple y con instrucciones fácilmente legibles, incluso por no expertos (ver figura 26). No se puede tener el control del programa en cuanto a tiempos de ejecución y control de registros bit a bit. La mayoría de compiladores para este lenguaje pueden utilizarse únicamente bajo ambiente Windows.

Figura 26. Ejemplo de lenguaje de programación BASIC.

```
'* Nombre del archivo: blink.bas
'* Ejemplo de programa que hace parpadear un LED
'* conectado al PORTB.0 utilizando Picbasic Pro.
'* El pic utilizado es el 16F628A de la gama media.
*****************************************************************

@ DEVICE PIC16F628A,WDT_OFF,PWRT_ON,MCLR_ON,BOD_OFF,CPD_OFF,PROTECT_OFF

Define OSC 4

loop: High PORTB.0          ' Encienda LED
      Pause 200             ' Retardo de 200 ms
      Low PORTB.0           ' apague LED
      Pause 200             ' retardo de 200 ms
Goto loop                   ' Ejecute este ciclo repetitivamente

End
```

Fuente: http://electronicadesarrollo.blogspot.com/2007/12/programacin-de-pics.html

2.3.5 Herramientas de desarrollo. Las herramientas de desarrollo están formadas por un conjunto de programas e interfaces que permiten realizar los proyectos de la forma más eficiente posible.

Las principales herramientas de ayuda al desarrollo de sistemas basados en Microcontroladores se describen a continuación:

Compilador. La programación en un lenguaje de alto nivel (como C o Basic) permite disminuir el tiempo de desarrollo de un producto y si además está familiarizado con C o Basic es una buena opción. No obstante, cuando el compilador convierta el código del programa a un lenguaje ensamblado, cada línea de código del programa en lenguaje de alto nivel habrá generado bastantes más líneas de código en lenguaje ensamblador, normalmente en una relación de uno a tres. Esto significa que para utilizar un lenguaje de alto nivel necesitaremos un Microcontrolador con una capacidad de memoria relativamente grande.

Si el programa que estamos desarrollando necesita utilizar números con decimales, o con notación científica o se utilizan operaciones complejas, como pueden ser las trigonométricas, es casi obligado utilizar un lenguaje de alto nivel. Pero si lo que se va a hacer es manipular bits en registros, entradas, salidas y cálculos sencillos, el lenguaje ensamblado es la mejor opción.

Simulador. Se trata de software que es capaz de ejecutar en un PC programas realizados para el Microcontrolador. Los simuladores permiten tener un control absoluto sobre la ejecución de un programa, siendo ideales para la depuración de los mismos. Su gran inconveniente es que es difícil simular la entrada y salida de datos del Microcontrolador. Tampoco cuentan con los posibles ruidos en las entradas, pero, al menos, permiten el paso físico de la implementación de un modo más seguro y menos costoso, puesto que ahorraremos en grabaciones de chips para la prueba in-situ.

Placas de evaluación. Se trata de pequeños sistemas con un Microcontrolador ya montado y que suelen conectarse a un PC desde el que se cargan los programas que se ejecutan en el Microcontrolador. Las placas suelen incluir visualizadores LCD, teclados, LEDs, fácil acceso a los pines de E/S, etc. Pueden incluir un programa de control o sistema operativo que recibe el nombre de programa monitor. El programa monitor de algunas placas de evaluación, aparte de permitir cargar programas y datos en la memoria del Microcontrolador, puede permitir en cualquier momento realizar ejecución paso a paso, monitorizar el estado del Microcontrolador o modificar los valores almacenados los registros o en la memoria.

Programador. Es un dispositivo que conectado a un PC permite grabar en el Microcontrolador el programa desarrollado. Algunos puede fabricarlos uno mismo y resultan muy económicos. También existe software gratuito para programar no ya solo Microcontroladores sino también otros dispositivos, como memorias.

Actualmente se tiende a realizar la programación en la propia placa de utilización mediante ISP, In System Programmation o ICSP, In Circuit Serial Programation. De esta manera se puede programar al Microcontrolador una vez esté montado en la placa del circuito utilizando una conexión de dos, tres o cuatro terminales. Para utilizar esta técnica se utiliza un programador que suele ser muy sencillo y que en algunos casos puede construir uno mismo.

Paquetes IDE. Actualmente existen paquetes de software denominados "Entornos de Desarrollo Integrado", IDE, que suelen funcionar bajo Windows y que incluyen editores de texto para el ensamblador o el compilador, permiten la simulación del programa y también pueden integrar el control de emuladores y programadores de dispositivos.

3. DISEÑO DEL BANCO DE EXPERIMENTACIÓN

3.1 INTRODUCCIÓN

Este proyecto busca brindar herramientas para el proceso de enseñanza y aprendizaje de la asignatura Potencia Fluida e Ingeniería de Control, mediante el diseño y construcción de un banco de experimentación por medio del cual los estudiantes podrán actualizar los conocimientos sobre el uso de sistemas de control en unidades de potencia hidráulica. El sistema de control está basado en un Controlador lógico programable KOYO DL06 DD1, y una tarjeta Arduino Mega que tiene un Microcontrolador ATMET ATMEGA.

Para el montaje del sistema de control fué necesario diseñar y una estructura (ver figura 61) que cumpliera con los requerimientos de funcionalidad, versatilidad, espaciamiento, entre otros. Se diseñó un tablero de potencia, para proporcionar las tensiones necesarias para el funcionamiento del banco (ver figura 102).

3.2 IDENTIFICACIÓN Y SELECCIÓN DE LOS COMPONENTES

Para cumplir con las principales características tenidas en cuenta en la matriz de calidad que se desarrolló para cumplir con las necesidades del usuario, tales como modularidad, seguridad, distribución de espacio, entre otras, se realizó una clasificación y descripción de los distintos componentes utilizados de la siguiente manera:

- Componentes de control.
- Componentes de protección y potencia.
- Componentes de maniobra.

3.2.1 Componentes de control. Dispositivos que se utilizan para gobernar el control de un sistema.

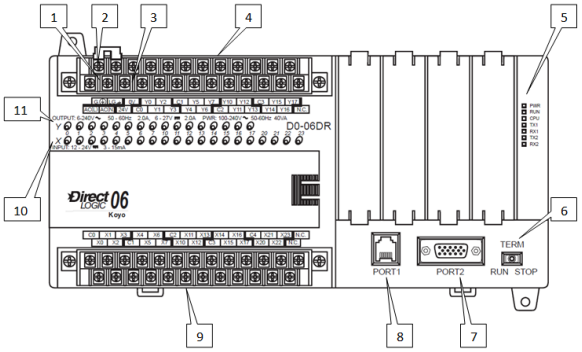
3.2.1.1 Controlador Lógico programable (PLC). Teniendo en cuenta los parámetros a controlar la cantidad de y origen de las señales a controlar se seleccionó un PLC KOYO DL06 DD1 (ver figura 27), el cual cumple con la norma IEC 601131[8].

Figura 27. PLC KOYO DL06 DD1.

Fuente: Autores.

En la figura 28 y en la tabla 2, se identifican las principales partes del PLC KOYO DL06 DD1.

Figura 28. Identificación de las principales partes del PLC KOYO DL06 DD1.



Fuente: Autores.

[8] Norma IEC 601131 define los estándares de leguajes de programación para los Autómatas Programables.

Tabla 2. Identificación de las principales partes del PLC KOYO DL06 DD1.

1	Alimentación con corriente alterna de 110 o 220V AC
2	Conexión para Tierra
3	Fuente interna de 24V DC hasta 350 mA
4	Señales de salida (Yi) 10 a 24V DC / 1mA Max
5	Leds indicadores de estado
6	Modo de funcionamiento del equipo
7	Puerto de comunicación 2
8	Puerto de comunicación 1
9	Señales de entrada (Xi) 4 mA @ 10V DC / 8,5 mA @ 24V DC
10	Indicadores de estado de entrada
11	Indicadores de estado de salida

Fuente: Autores.

El PLC KOYO DL06 DD1 Tiene 14,8K palabras de memoria, con 7,6K palabras de memoria Ladder y 7,6K palabras de memoria V para usuario (registros de datos). El almacenamiento del programa se hace en una memoria Flash, que es una parte de la CPU en el PLC.

Tiene cuatro ranuras para colocar módulos opcionales de I/O; Tiene incorporado dos puertos de comunicación, de modo que puede conectarlo fácilmente a un programador portátil, a una interface de operador o a una computadora personal sin necesitar de ningún hardware adicional e incluso puede crear redes seriales para aumentar el número de entradas y salidas. Con un módulo opcional puede crear una red de comunicación Ethernet con otros aparatos.[9]

Leds indicadores de estado. Indican el estado en que se encuentra el equipo. En la tabla 3 se puede observar el significado de cada LED de estado.

[9] Fuente: Automation Direct. manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 31.

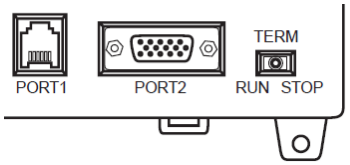
Tabla 3. Descripción de los LED de estado.

Indicador	Estado	Significado
PWR	ON	Alimentación en orden
	OFF	La alimentación ha fallado
RUN	ON	La CPU está en modo RUN
	OFF	La CPU está en modo STOP o PROGRAM
	Parpadeando	La CPU está en modo de actualización de firmware
CPU	ON	Error por diagnóstico de la CPU
	OFF	Diagnóstico de la CPU en orden
	Parpadeando	Batería con bajo voltaje
TX1	ON	Están siendo transmitidos datos por la CPU: puerto 1
	OFF	No están siendo transmitidos datos por la CPU: puerto 1
RX1	ON	Están siendo recibidos datos por la CPU: puerto 1
	OFF	No están siendo recibidos datos por la CPU: puerto 1
TX2	ON	Están siendo transmitidos datos por la CPU: puerto 2
	OFF	No están siendo transmitidos datos por la CPU: puerto 2
RX2	ON	Están siendo recibidos datos por la CPU: puerto 2
	OFF	No están siendo recibidos datos por la CPU: puerto 2

Fuente: Autores.

Modos del equipo. El PLC tiene una palanca (ver figura 29), que permite seleccionar el estado del equipo, en la mitad se encuentra en el estado *TERM*, hacia la izquierda están el modo *RUN*, y hacia la derecha el modo *STOP*.

Figura 29. Palanca para cambiar modo del equipo.



Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 35.

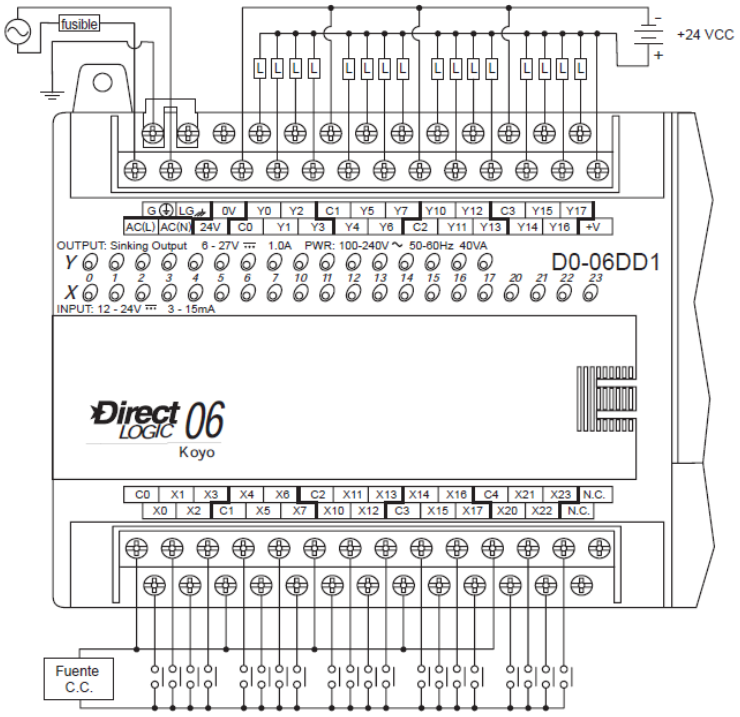
Modo PROGRAM (TERM). Permite entrar o modificar un programa en el PLC. En este modo no se ejecuta el programa de uso, ni se actualizan salidas.

Modo RUN. En este modo la CPU del PLC ejecuta el programa. Se pueden hacer cambios en temporizadores, memorias, pero no permite cambios de entradas y salidas.

Modo STOP. Se utiliza para detener la ejecución del programa.

Cableado. En la figura 30 se puede observar el método de cableado tanto de la alimentación externa como de las entradas y salidas del equipo.

Figura 30. Cableado de la alimentación externa, señales de entrada y salida.

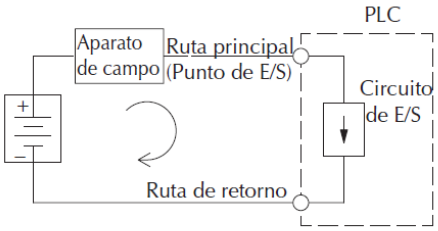


Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 70.

Alimentación externa. El PLC DO DL06 DD1 se alimenta externamente con una fuente de 110VAC.

Terminales "Comunes". Para que un circuito de entradas y salidas del PLC funcione, la corriente debe tener un camino de regreso que cierre el circuito, tal como lo muestra la figura 31.

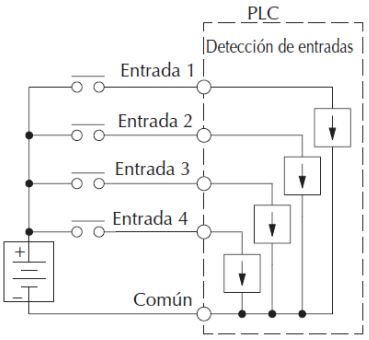
Figura 31. Ruta de la corriente.



Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 65.

Para disminuir la cantidad de terminales en el PLC, cuatro terminales principales (las entradas y salidas Xi y Yi son la ruta principales) se agrupan y comparten un común Ci como se observa en la figura 32, de esta manera solo son necesarios 5 terminales en vez de 8, para completar 4 circuitos.

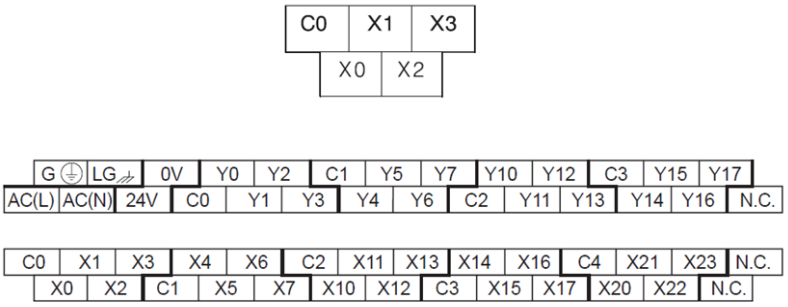
Figura 32. Terminales comunes.



Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 62.

En la rotula del PLC (ver figura 33), se pueden observar estos comunes, note que X0, X1, X2, X3, comparten un común que es C0.

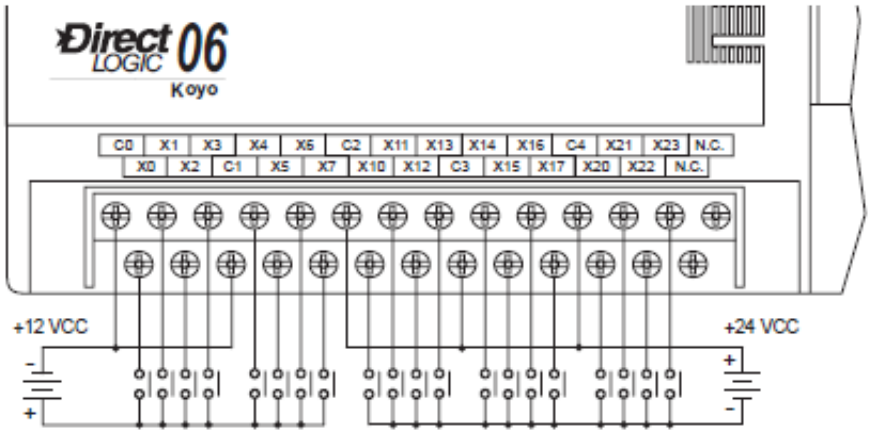
Figura 33. Rotula de entradas y salidas del PLC KOYO DL06 DD1.



Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 40.

Entradas y salidas. El PLC DO DL06 DD1 tiene veinte entradas y dieciséis salidas que soportan tensiones de 12 VDC a 24 VDC. Las entradas del PLC PLC se pueden conectar como drenadoras (NPN) o surtidoras (PNP). Tal como se observa en la figura 34.

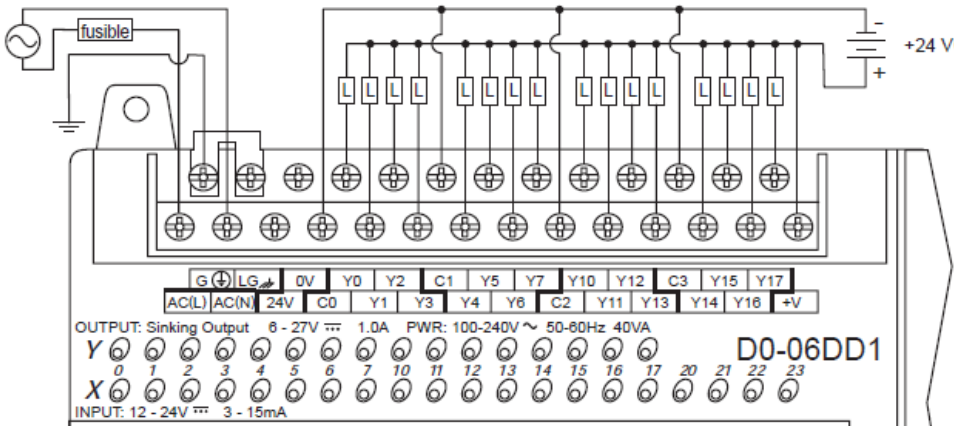
Figura 34. Equema de las conexiones de las entradas del PLC KOYO DL06 DD1.



Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 36.

Las salidas se pueden conectar únicamente como drenadoras (NPN), tal como se muestra en la figura 35.

Figura 35. Equema de las conexiones de las salidas del PLC KOYO DL06 DD1.



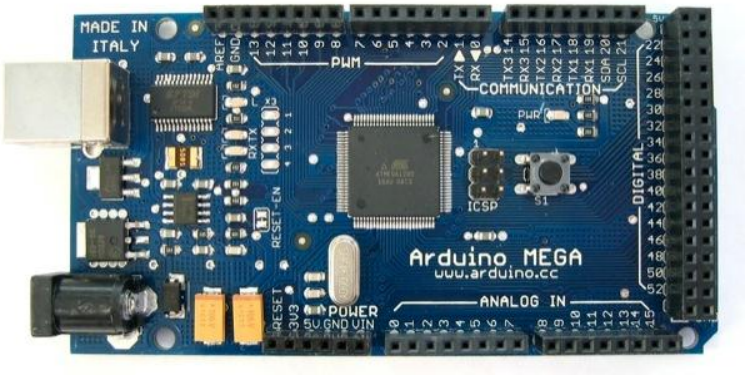
Fuente: Automation Direct. Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 42.

Comunicación. Tiene dos puertos de comunicación seriales incorporados. El puerto 1 (RS232C solamente) se utiliza generalmente para conectar con un programador D2-HPP, DirectSOFT, una interface de operador, un esclavo MODBUS o DirectNET solamente. The baud rate es fijo en 9600 baud en el puerto 1. El puerto 2 (RS232C/RS422/RS485) se puede usar para conectar con un D2-HPP, DirectSOFT, una interface del operador, un maestro o esclavo MODBUS RTU o DirectNET o ASCII como entrada y salida.

Programación. La programación del equipo se realizar por medio del programa *DirectSOFT* V5, el cual utiliza un lenguaje de programación Ladder, Nemónico o Diagrama de bloques.

3.2.1.2 Tarjeta de control ARDUINO MEGA. El otro sistema de control alternativo al PLC, consiste en una tarjeta de control Arduino Mega que tiene incorporado un Microncontrolador ATMEL ATMEGA 1280K esta se puede observar en la figura 36.

Figura 36. Tarjeta de control ARDUINO MEGA.



Fuente: http://arduino.cc/es/Main/ArduinoBoardMega

Descripción general[10]. Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar para hacer que los ordenadores puedan sentir y controlar el mundo físico a través de un ordenador personal. Es una plataforma de desarrollo de computación física (physical computing) de código abierto, basada en una placa con un sencillo microcontrolador y un entorno de desarrollo para crear software (programas) para la placa. Arduino está basado en los Microcontroladores ATMEGA168, ATMEGA328, ATMEGA1280 y ATMEGA2560.

Se puede usar Arduino para crear objetos interactivos, leyendo datos de una gran variedad de interruptores y sensores y controlar multitud de tipos de luces, motores y otros actuadores físicos. Los proyectos de Arduino pueden ser autónomos o comunicarse con un programa (software) que se ejecute en tu

[10] Rafael Enríquez Herrador; Guía de Usuario de Arduino. Universidad de Córdoba. 13 de noviembre de 2009. p. 35.

ordenador (ej. Flash, Processing, MaxMSP). La placa puede montarse o comprarla ya lista para usar, El software Arduino está publicado bajo una licencia libre. El lenguaje puede ampliarse a través de librerías de C++ que pueden ser creadas por los usuarios o descargadas de internet.

El lenguaje de programación de Arduino es una implementación de Wiring, una plataforma de computación física parecida, que a su vez se basa en Processing, un entorno de programación multimedia.

Principales Características. En la tabla 4 se describen las principales características de la plataforma Arduino.

Tabla 4. Características de la plataforma Arduino.

Microcontrolador	ATmega1280
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	54 (14 proporcionan salida PWM)
Pines de entrada analógica	16
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	128 KB de las cuales 4 KB las usa el gestor de arranque (bootloader)
SRAM	8KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Fuente: Autores

Alimentación. El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.

Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería pueden conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER).

La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan más de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

VIN. La entrada de voltaje a la placa Arduino cuando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.

5V. la fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.

3V3. Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.
GND. Pines de toma de tierra.

Memoria. El ATmega1280 tiene 128KB de memoria flash para almacenar código (4KB son usados para el arranque del sistema (bootloader).El ATmega1280 tiene 8 KB de memoria SRAM. El ATmega1280 tiene 4KB de EEPROM, a la cual se puede acceder para leer o escribir con la [Reference/EEPROM |librería EEPROM]].

Pines de la tarjeta:

Entradas y Salidas digitales. Cada uno de los 54 pines digitales en el Arduino Atmega 1280 pueden utilizarse como entradas o como salidas usando las funciones pinMode(), digitalWrite(), y digitalRead(). Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad máxima de 40mA y tiene una resistencia interna (desconectada por defecto) de 20-50kOhms.

Interrupciones Externas. Estos pines se pueden configurar para lanzar una interrupción en un valor LOW (0V), en flancos de subida o bajada (cambio de LOW a HIGH (5V) o viceversa), o en cambios de valor, la tabla 5 muestra los puertos de interrupción que tiene el Arduino.

Tabla 5. Puertos de interrupción.

interrupción 0	pin 2
interrupción 1	pin 3
interrupción 2	pin 21
interrupción 3	pin 20
interrupción 4	pin 19
interrupción 5	pin 18

Fuente: Autores.

PWM: de 0 a 13. Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a través de la función analogWrite().

Comunicación SPI. Estos pines proporcionan comunicación SPI, que a pesar de que el hardware la proporcione actualmente no está incluido en el lenguaje Arduino. Pin 50 (SS), pin 51 (MOSI), pin 52 (MISO), pin 53 (SCK).

LED 13. Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

Entradas Analógicas. El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de tierra a 5 voltios, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference().

AREF. Voltaje de referencia para las entradas analógicas. Usado por analogReference().

Reset. Suministrar un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los programas que no dejan acceso a este botón en la placa.

Comunicaciones. EL Arduino Mega facilita en varios aspectos la comunicación con el ordenador, otro Arduino u otros Microcontroladores. El ATmega1280 proporciona cuatro puertos de comunicación vía serie UART TTL (5V). Un chip integrado en la placa canaliza esta comunicación serie a través del USB y los drivers FTDI (incluidos en el software de Arduino) proporcionan un puerto serie virtual en el ordenador.

El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDS RX y TX de la placa parpadearan cuando se detecte comunicación transmitida través del chip y la conexión USB (no parpadearan si se usa la comunicación serie a través de los pines 0 y 1).

La librería SoftwareSerial permite comunicación serie por cualquier par de pines digitales del Mega. El ATmega1280 también soporta la comunicación I2C (TWI) y SPI. El software de Arduino incluye una librería Wire para simplificar el uso el bus I2C.

Programación. La programación del equipo se realiza mediante el software Arduino, el cual maneja un lenguaje C++.

3.2.2 Componentes de protección y potencia. Los componentes de protección se encargar de salvaguardar el sistema en caso de alguna anormalidad. Los componentes de potencia se encargan de distribuir las tensiones necesarias para el funcionamiento del sistema.

3.2.2.1 Fusibles de acción rápida. Estos fusibles se utilizaran para la protección de las líneas de tensión que alimentan los componentes del sistema de control.

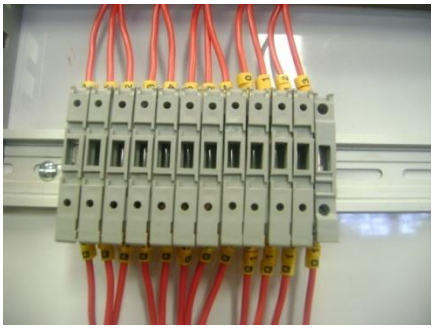
Se utilizó un fusible de acción rápida marca CHINT como el que se ilustra en la figura 37, para la fase principal del banco de experimentación, su corriente de accionamiento es de 16Amp. También se utilizó un fusible de acción rápida marca Schneider en la Fase del PLC cuya corriente de accionamiento es de 1Amp y otro en el tablero de potencia marca Sanjin cuya corriente de accionamiento es de 20 Amp.

Figura 37. Fusible de acción rápida.

Fuente: http://www.chint.es/Spanish/bigjpg/Products/MCB/UB.JPG

3.2.2.2 Fusibles de fusión rápida para salidas del PLC. El PLC no tiene fusibles internos en la salida, por lo tanto el fabricante recomienda colocar fusibles de fusión rápida para proteger los circuitos de salida que están sometidos a tensiones de 24VDC. Se instalaron en cada una de las salidas del PLC bornes protafusibles como el que se ilustra en la figura 38.

Figura 38. Borne porta fusible.



Fuente: Autores.

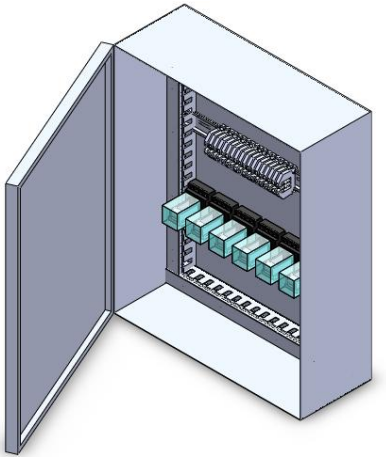
3.2.2.3 Alarma. Se instaló una alarma tipo campana marca Vera (ver figura 39) de 110-150 VAC, la cual es accionada por el PLC y el Microcontrolador de acuerdo a su programación.

Figura 39. Alarma.

Fuente: Autores.

3.2.2.4 Tablero de potencia. Este tablero está ubicado en la pared del laboratorio, su función es alojar los relevos para el aislamiento de la línea de neutro de las electroválvulas y los relevos encargados del accionamiento del motor eléctrico. Este se puede observar en la figura 40.

Figura 40. Tablero de potencia.



Fuente: Autores.

3.2.2.5 Bornes. Se utilizan bornes como los ilustrados en la figura 41, para las múltiples conexiones eléctricas que se requieren, además de lo anterior también cumplen la función de dar orden al sistema de cableado. Se utilizaron bornes de 40mm.

Figura 41. Bornes.

Fuente: http://www.terminalblock.es/2-1-screw-terminal-block.html

3.2.2.6 Selección de cables. Estos fueron seleccionados de acuerdo a la corriente que manejan los diferentes componentes del sistema.
De a cuerdo a la tabla 6 y 7 se seleccionó el cableado de la siguiente manera:

- Cable TFF 18 AWG 600V 60 C color rojo y negro, para las conexiones de 24 VDC, 12VDC, 110AC electroválvulas, PLC, relevos

- Cable TWK 14 AWG 600V 60 C rojo y negro, para las conexiones de 110 VAC del arranque del motor principal

- Triplex ST encauchetado calibre 12, Para la alimentación principal del banco de experimentación.

Tabla 6. Características de cables tipo TFF y TWK flexibles de cobre Centelsa.

Tipo	1. Conductor		2. Aislamiento Espesor	Resistencia DC a 20°C	Diámetro Exterior	Peso Total Aproximado	Capacidad de Corriente (*)
	Calibre	Diámetro					
	AWG	mm	mm	Ohm/km	mm	kg/km	A
TFF	18	1,17	0,76	21,423	2,77	15	10
TFF	16	1,48	0,76	13,452	3,08	20	13
TWK	14	1,89	0,76	8,444	3,49	29	18
TWK	12	2,38	0,76	5,315	3,98	42	25
TWK	10	2,99	0,76	3,344	4,59	62	30
TWK	8	3,75	1,14	2,102	6,13	102	40
TWK	6	4,75	1,52	1,323	7,91	166	55

Notas

Los datos aquí indicados están sujetos a las tolerancias normales de fabricación y pueden ser modificados sin previo aviso.

(*) Capacidades de corriente a temperatura ambiente 30ºC y temperatura conductor 60ºC. Norma base NTC 2050 sección 400.

Otras configuraciones, calibres y colores no especificados en este catálogo están disponibles bajo pedido.

Fuente:http://www.centelsa.com.co/index.php?p=productos.vsitemview&itemact=main

Tabla 7. Características de cables tipo ST flexibles de cobre Centelsa.

1. Conductor		Número de Conductores	Diámetro del Núcleo	Diámetro Exterior	Peso Total Aproximado	Resistencia DC a 20°C	Capacidad de Corriente (A)	
Calibre	Diámetro							
AWG	mm	No	mm	mm	kg/km	Ohm/Km	(*)	(**)
18		2	2,34	6,66	62	21,42	11	8
	1,17	3	2,52	7,05	74			
		4	2,82	7,71	90			
16		2	2,96	7,28	79	13,45	15	11
	1,48	3	3,19	7,72	96			
		4	3,57	8,46	117			
14		2	3,78	8,10	105	8,44	22	17
	1,89	3	4,07	8,60	130			
		4	4,56	9,45	160			
12		2	4,76	9,08	142	5,31	28	22
	2,38	3	5,13	9,66	178			
		4	5,75	11,41	241			
10		2	5,98	10,30	197	3,34	35	27
	2,99	3	6,44	11,75	271			
		4	7,22	12,88	336			
8		2	7,50	14,90	372	2,10	45	34
	3,75	3	8,08	15,81	462			
		4	9,05	18,35	612			
6		2	9,50	18,44	576	1,32	65	50
	4,75	3	10,24	19,55	717			
		4	11,47	21,39	886			

Notas

Los datos aquí indicados están sujetos a las tolerancias normales de fabricación y pueden ser modificados sin previo aviso.

Capacidades de corriente a temperatura ambiente 30°C y temperatura conductor 90ºC. Norma base NTC 2050 Sección 400.

(*) Hasta tres conductores transportando corriente.

(**) Cuatro conductores transportando corriente.

Otras configuraciones, calibres y colores no especificadas en este catálogo están disponibles bajo pedido.

Fuente:http://www.centelsa.com.co/index.php?p=productos.vsitemview&itemact=main

3.2.3 Componentes de maniobra. Su función es manipular las condiciones de un determinado circuito.

3.2.3.1 Selector. Se utilizó un selector de dos posiciones como el ilustrado en la figura 42 para energizar el banco, otro de tres posiciones para seleccionar las tensiones hacia el PLC o el Microcontrolador.

Figura 42. Selector.

Fuente: http://casalemana.com/UserFiles/image/Productos

3.2.3.2 Parada de emergencia. Según la norma IEC 241-1, toda máquina industrial debe tener mecanismos para detenerla de una forma rápida en caso de peligro, para ello se utilizó un pulsador de parada de emergencia (ver figura 43). Este tiene las siguientes características:

- Es de color rojo.
- Está ubicado en un lugar estratégico y de fácil acceso.
- Tiene enclavamiento mecánico.
- Tiene un área de accionamiento de 40mm

Figura 43. Pulsador de parada de emergencia.

Fuente: http://www.chint.es/product.asp?classid=20&id=15

3.2.3.3 Pulsadores. Se utilizaron pulsadores para establecer comunicación entre el operario y el sistema de control, estos accionan componentes como el motor, reset PLC.

En la tabla 8 se muestra la norma DIN EN 60204-1 la cual establece el código de colores para los pulsadores.

Tabla 8. Resumen norma DIN EN 60204-1.

Color	Significado
	Seguro, arranque
	Anormal
	Emergencia, parada normal
	Obligatorio
	No tiene significado especifico

Fuente: Autores.

Se utilizaron los siguientes pulsadores:

Verde:

- Energía principal del sistema de control.

- Inicio motor principal.
- Reset Microncontrolador.
- Reset PLC.

Rojo:

- Parada del motor

La figura 44 muestra el tipo de pulsadores utilizados.

Figura 44. Pulsador.

Fuente: http://casalemana.com

3.2.3.4 Pilotos luminosos. En la tabla 9 se muestra la norma DIN EN 60204-1 establece el código de colores para los pilotos luminosos.

Tabla 9. Resumen norma DIN EN 60204-1.

Color	Significado
	Funcionamiento normal
	Anormal
	Emergencia, equipo detenido
	Obligatorio
	No tiene significado especifico

Fuente: Autores.

Se utilizaron los siguientes pilotos indicadores:

Verde:

- Energía principal del sistema de control.
- Motor en marcha.
- Primer nivel de presión
- Segundo nivel de presión
- Microntrolador funcionando.
- PLC en funcionamiento.

La figura 45 muestra el tipo de pilotos luminosos utilizados.

Figura 45. Piloto luminoso.

Fuente: http://www.chint.es/Spanish/bigjpg/Products/Pilot%20Device/NFM1.JPG

3.2.3.5 Relevadores. Se utilizan relés electromecánicos[11] de dos clases, unos para el aislamiento de la línea neutro de las electroválvulas, y otros para la tarjeta de potencia.

Los relés utilizados para el aislamiento línea de neutro de las electroválvulas se pueden observar en la figura 46. La tabla 10 muestra las principales característica de estos relevos.

[11] Los relés electromecánico consisten en una serie de contactos se sierran por medio campo magnético que se produce por la energización eléctrica su bobina.

Tabla 10. Características relevo de 110VAC.

MARCA	Kest
REFERENCIA	KIR3P11QX12
ALIMENTACIÓN BOBINA	12VDC
TENSIÓN MAX. DE CONMUTACIÓN	28VDC/240VAC
MAX. CONRRIENTE DE CONMUTACIÓN	15A @ 28VDC 10A @ 240VAC
MAX. CONMUTACIÓN DE ENERGÍA	240W/2.5KVA
RESISTENCIA DEL CONTACTO	180 Ω
NUMERO DE CONTACTOS	3NA/3NC

Fuente: Autores.

Figura 46. Relevador de 110 VAC.

Fuente: http://www.kestparts.com/components/miscellaneous/ relays/categories

Los relés utilizados para la tarjeta de potencia se pueden observar en la figura 47. La tabla 11muestra las principales características de estos relevos

Tabla 11. Características del relevo de 24VDC.

MARCA	LIMING
REFERENCIA	JZC-22F
ALIMENTACIÓN BOBIA	3 – 48VDC
MAX. TENSIÓN DE CONMUTACIÓN	24VDC/250VAC
MAX. CONRRIENTE DE CONMUTACIÓN	15A @ 28VDC 10A @ 240VAC
MAX. CONMUTACIÓN DE ENERGÍA	240W/2.5KVA
RESISTENCIA DEL CONTACTO	100mΩ
NUMERO DE CONTACTOS	1NA/1NC

Fuente: Autores.

Figura 47. Relevador de 24VDC.

Fuente: Autores.

3.2.3.6 Sensores inductivos. Las señales que llegan al sistema de control provenientes del banco hidráulico las proporcionan ocho sensores inductivos como los ilustrados en la figura 48, los cuales están ubicados al inicio y al final de los actuadores. La tabla 12 muestra las principales característica de estos sensores.

Figura 48. Sensor inductivo.

Fuente: Autores.

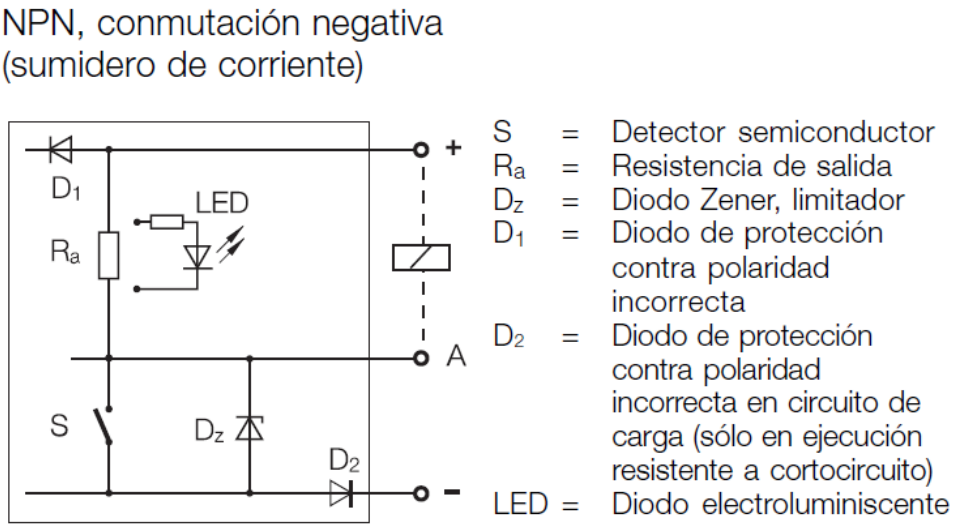
Tabla 12. Características de los sensores inductivos.

MARCA	CONCH
REFERENCIA	QS-1708NA
ALIMENTACIÓN	10 – 30 VDC
MAX. RANGO DE OPERACIÓN	8mm ± 10%
RANGO OPTIMO DE DETECCIÓN	0 A 7mm
CORRIENTE DE CONSUMO	10mA O MENOS
SEÑAL	NPN NORMALMENTE ABIERTO
CORRIENTE DE SEÑAL	MAX. 150mA
NÚMERO DE HILOS	3
MAX FRECUENCIA DE LA SAÑAL	200HZ
GRADO DE PROTEXIÓN	IP67
RANGO DE TEMPEATURA/ HUMEDAD	-20 A +70C/ 35 A 95%
MATERIAL	ABS
PESO	49g

Fuente. Autores.

En la figura 49 se ilustra el esquema eléctrico del sensor inductivo tipo NPN de tres hilos normalmente abierto.

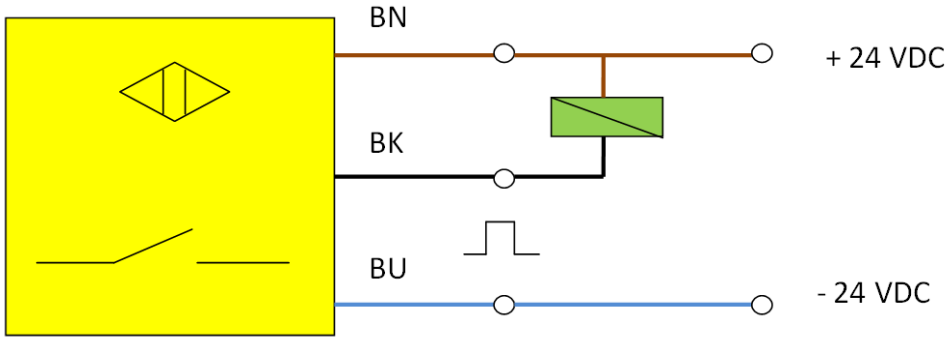
Figura 49. Esquema electrico del sensor inductivo.



Fuente: http://www.nortecnica.com.ar/pdf/teoria_inductivos_2_2.pdf

La figura 50 ilustra un esquema gráfico de las conexiones del sensor, se puede observar que se alimenta por medio de las conexiones BN y BU y la señal es la conexión BK.

Figura 50. Esquema grafico de las conexiones del sensor.



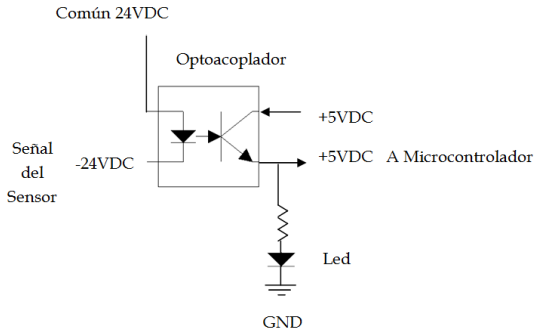
Fuente: Autores.

3.2.3.7 Tarjeta de Interfaz para señales de entrada de la tarjeta de control.

Los sensores inductivos producen señales de 24VDC, estas señales son las que sistema de control debe interpretar para posteriormente controlar el sistema hidráulico. En el caso de la tarjeta control Arduino Mega, esta soporta tensiones en su entrada de 5 VDC, por lo tanto fue necesario diseñar una tarjeta que redujera las tensiones de 24VDC a 5VDC.

El circuito consiste básicamente de Optoacopladores los cuales son activados por las señales de -24VDC proveniente de los sensores y 24VDC de un común y conmuta las señales de 5VDC hacia la Tarjeta de control, en la figura 51 se puede observar un esquema básico de esta tarjeta.

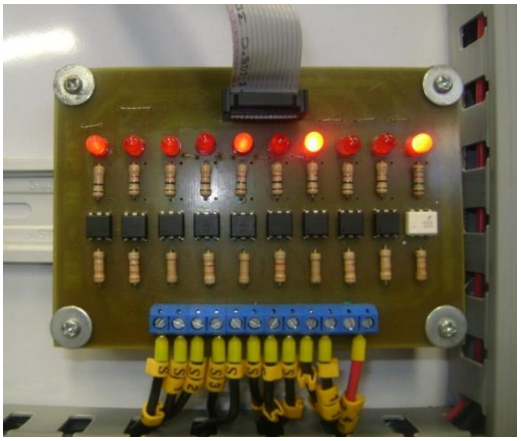
Figura 51. Esquema básico de la tarjeta de interfaz de la señales de entrada.



Fuente: Autores.

La tarjeta tiene unos Led indicadores de señal como se puede observar en la figura 52 los cuales sirven para visualizar el estado de las señales de entrada.

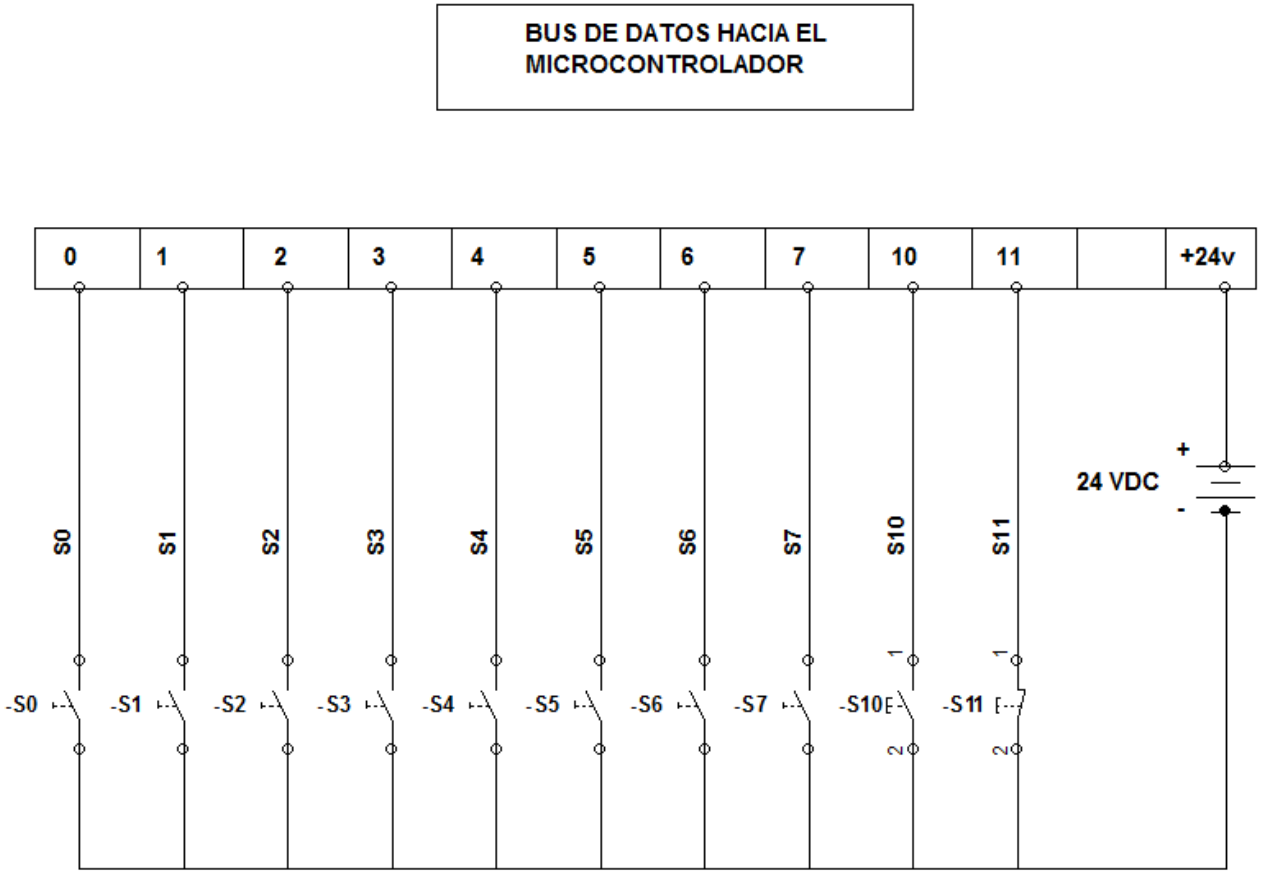
Figura 52. Tarjeta de interfaz para señales de entrada.



Fuente: Autores.

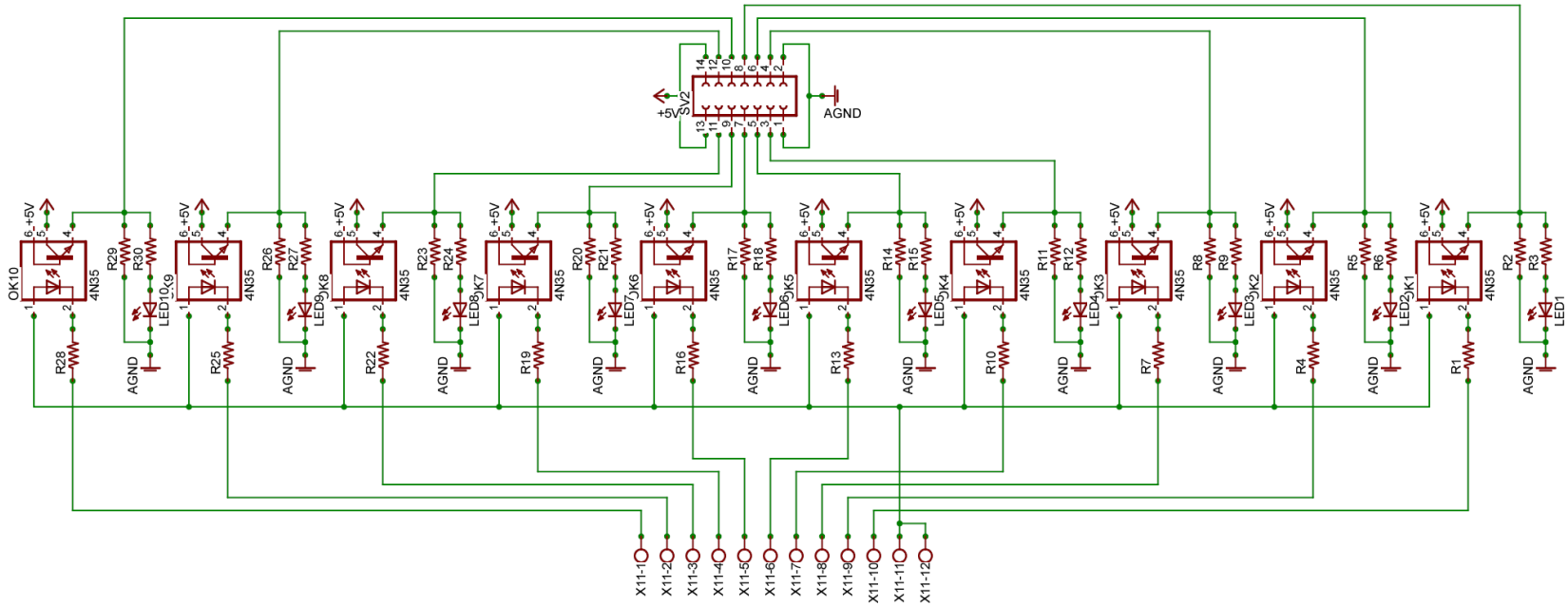
En la figura 53 se ilustra las conexiones electricas de la tarjeta de interface para las señales de entrada, las señales provenientes de los sensores llegan a la tarjeta y por medio de un bus de datos las transmite al microcontrolador. En la figura 54 se puede observar el diseño del circulito eléctrico de la tarjeta de interface para señales de entradas.

Figura 53. Esquema de las conexiones electricas en la tarjeta de interfaz para entrada de señales.



Fuente: Autores.

Figura 54. Esquema electrico de la tarjeta de interfaz para señales de entaradas.



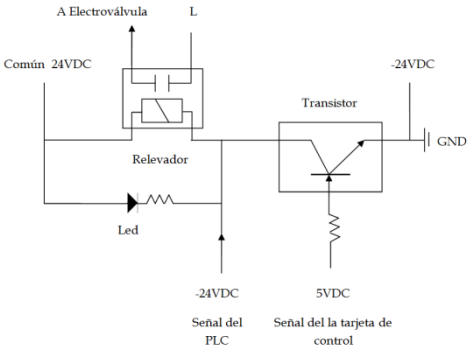
Fuente: Autores.

3.2.3.8 Tarjeta potencia. Las electroválvulas a controlar trabajan a tensiones de 120VAC, por lo tanto fué necesario diseñar una tarjeta para accionar estas electroválvulas a partir de las señales de 24VDC procedentes del PLC, y de 5VDC provenientes de la tarjeta de control.

La tarjeta consta de unos transistores que son activados por la señal de 5VDC proveniente de la tarjeta de control, esta conmutan la señal de -24VDC que acciona la bobina de 24VDC un relevo, el esquema basico de esta tarjeta se puede visualizar en la figura 55 y el diseño del circulto electrico se puede observar el la figura 58.

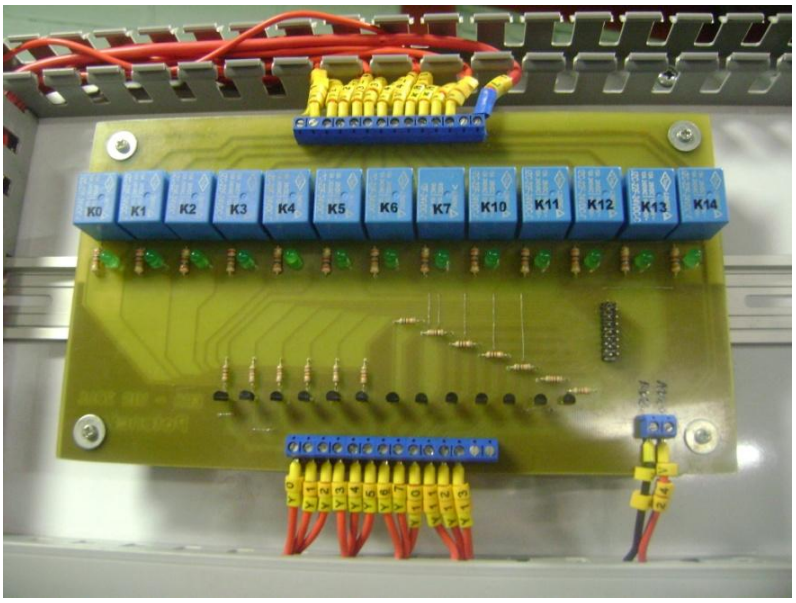
Cuenta también con una borneras disponibles para las señales de -24VDC provenienetes del PLC, cuando cualquera de los dos sistemas de control activa la señal de -24VDC, esta es dirigida a la Bobina de 24VDC del relevador, que activa sus conatacos conmutando la linea de fase de 120 VAC hacia las bobinas de las electroválvulas. Adicionalmente cuenta con unos Led indicadores de señal como se puede observar en la figura 56.

Figura 55. Esquema basico de la tarjeta de potencia.



Fuente: Autores.

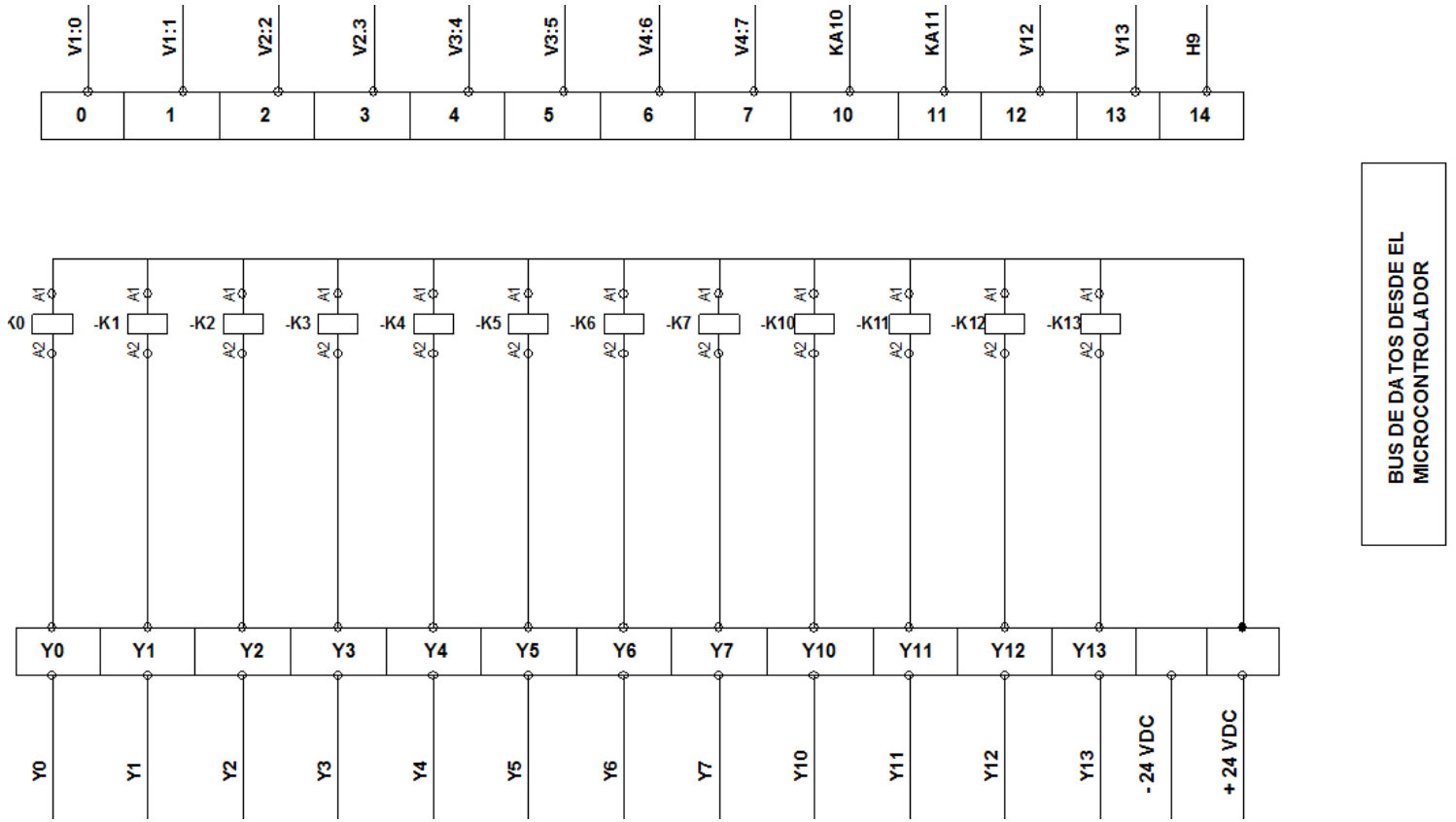
Figura 56. Tarjeta de potencia.



Fuente: Autores.

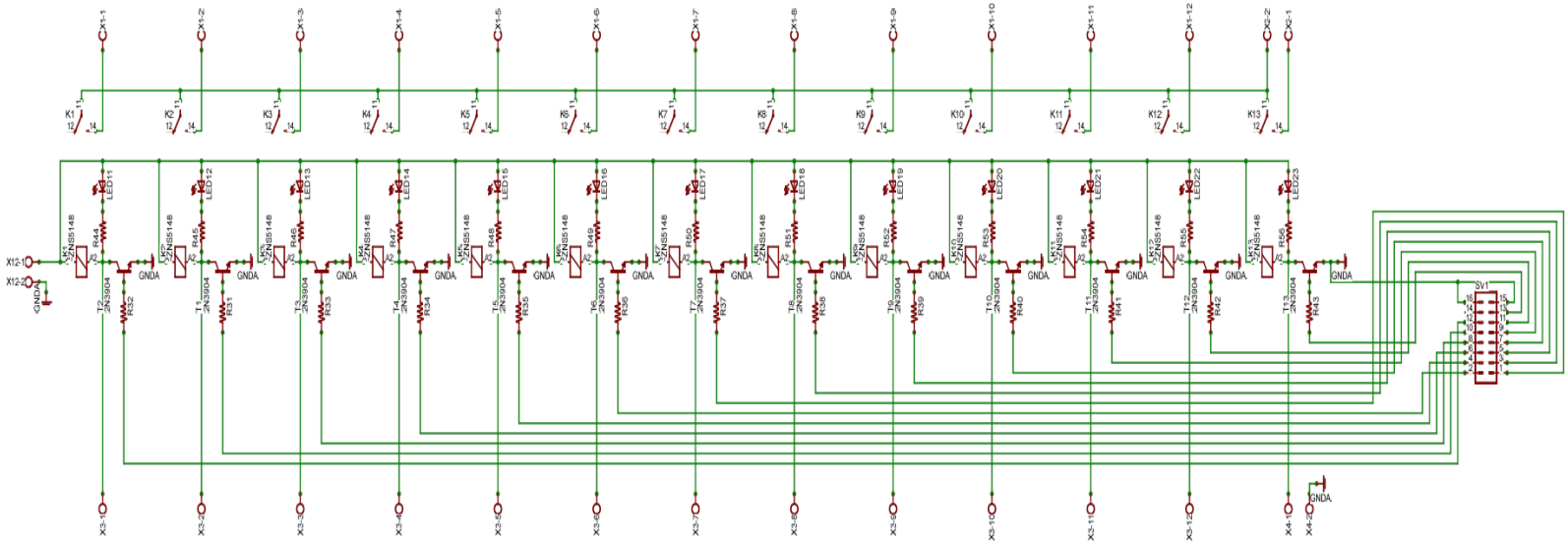
En la figura 57 se abservan las distintas conexiones que posse la tarjeta de potencia. Las señales provenientes del PLC se comunican con la targta por medio de bornes, se igal manera lo hacen la señales de salida. La comunicación con la tarjeta de control Arduino se realiza por medio de un Bus de datos.

Figura 57. Esquema de las conexiones eléctricas de la tarjeta de potencia.



Fuente: Autores.

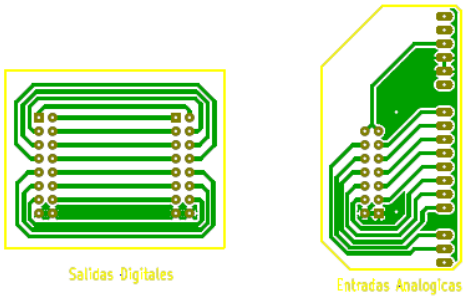
Figura 58. Diseñol del circuito electrico de la tarjeta de potencia.



Fuente: Autores.

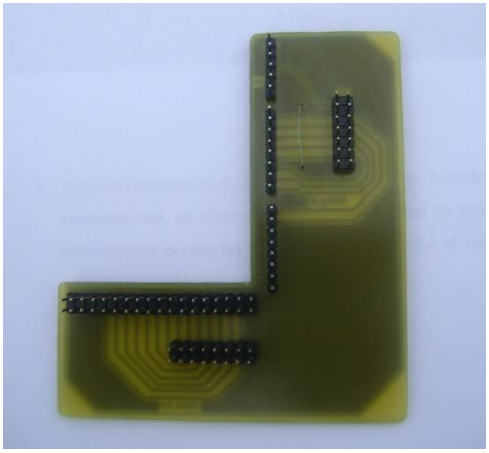
3.2.3.9 Tarjeta de periféricos auxiliares. Para comunicar tarjeta de interface para señales de entrada y la tarjeta de potencia con la tarjeta de control Arduino, se diseñó una tarjeta que consiste básicamente en unos puertos de bus de datos que comunican la tarjeta de control Arduino Mega con las tarjetas de interface de señales de entrada y la tarjeta de potencia, el diseño del circuito eléctrico se puede observar en la figura 59. A demás de las señales esta tarjeta también suministra las tensiones adicionales de 5VDC y GND requeridas para el funcionamiento de las tarjetas. La figura 60 ilustra la tarjeta de periféricos auxiliares.

Figura 59. Equema electrico de la tarjeta de periféricos auxiliares.



Fuente: Autores.

Figura 60. Tarjeta de periféricos auxiliares.



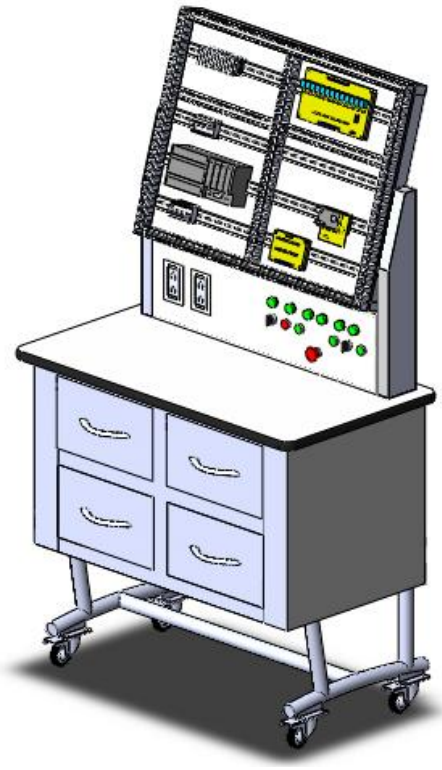
Fuente: Autores.

3.3 DISEÑO DE LA ESTRUCTURA.

El sistema de control está instalado sobre una estructura (ver figura 61), la cual debe cumplir con los siguientes requisitos:

- Debe ocupar un espacio moderado ya que el disponible en el laboratorio de Potencia Fluida es limitado.
- Se deben tener en cuenta criterios económicos para que sea viable su construcción.
- Diseño robusto debido al ambiente de trabajo al que estará sometido.
- Debido que al banco se deben conectar otros dispositivos, como computadores portátiles, debe contar con el espacio e instalaciones eléctricas para tal fin.
- Diseño estético.
- Fácil acceso a los dispositivos de control.

Figura 61. Estructura soporte sistema de control.



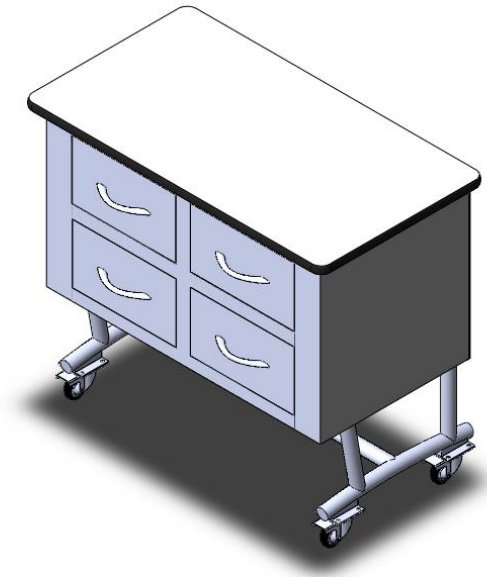
Fuente: Autores

La estructura está conformada básicamente por tres elementos:

- Mesa de trabajo.
- Panel para el soporte de los componentes de control.
- Tablero eléctrico.

3.3.1.1 Mesa de trabajo. Está provista de cuatro cajones para guardar los accesorios requeridos para las prácticas, además sobre ella se acopla el panel que soporta los componentes y el tablero eléctrico (ver figura 62). También está provista por ruedas con sistema de bloqueo, las cuales permiten su fácil desplazamiento.

Figura 62. Mesa de trabajo.



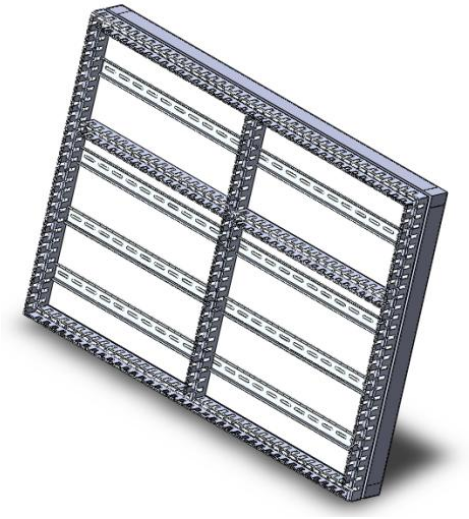
Fuente: Autores

3.3.1.2 Panel para el soporte de los componentes de control. Sobre este se instalan los componentes principales del sistema de control (ver figura 63), tales como:

- PLC.
- Microcontrolador.
- Tarjetas de interfaces de señales de entradas.

- Tarjeta de potencia.
- Bornes portafusibles.

Figura 63. Panel para el soporte de los componentes de control.

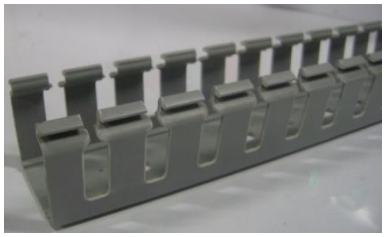


Fuente: Autores.

Tiene las siguientes dimensiones 82cm de largo por 60cm de alto y 9cm de espesor, la parte frontal está cubierta por un panel de formica, además cuenta con canaletas ranuradas para alojar el cableado y rieles Din para el montaje de los componentes.

3.3.1.3 Canaleta ranurada. Se seleccionaron canaletas ranuradas de 25x40mm marca DEXSON ref 3CARGR25x40 (ver figura 64).

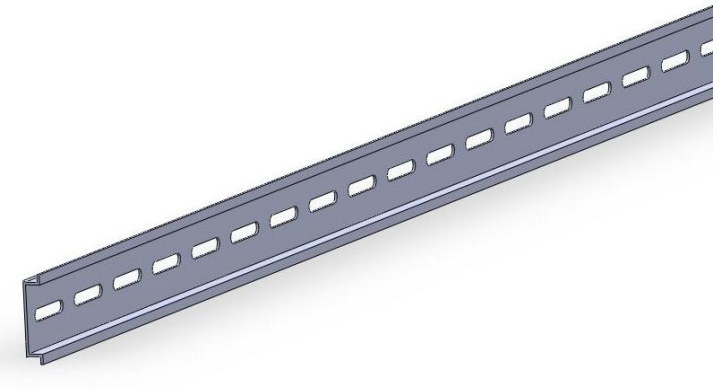
Figura 64. Canaleta ranurada.



Fuente: Autores.

3.3.1.4 Riel Din. Con el objetivo de estandarizar el montaje de los componentes, se utilizaron rieles DIN en aluminio marca OMEGA (ver figura 65), los cuales cumplen con el estándar DIN EN 50022.

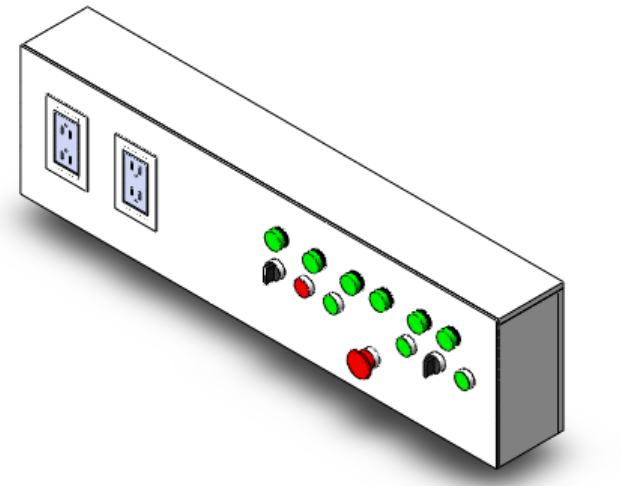
Figura 65. Riel DIN en aluminio.



Fuente: Autores.

3.3.1.5 Tablero de eléctrico. Este aloja la instalación eléctrica requerida para el funcionamiento del banco, además brida el espacio suficiente para la instalación de los comandos del sistema de control, como parada de emergencia, selectores, pilotos, pulsadores (ver figura 66).

Figura 66. Tablero eléctrico.



Fuente: Autores.

3.3.1.6 Tablero de mando. El tablero de mando se diseñó de acuerdo a la norma DIN EN 60204-1 la cual establece los colores para los pulsadores y pilotos. Este se encuentra instalado en tablero eléctrico. En la figura 67 se pueden apreciar las zonas en las que se divide el tablero de mando.

- Zona 1. Contiene el selector de energía para el banco y su respectivo piloto.

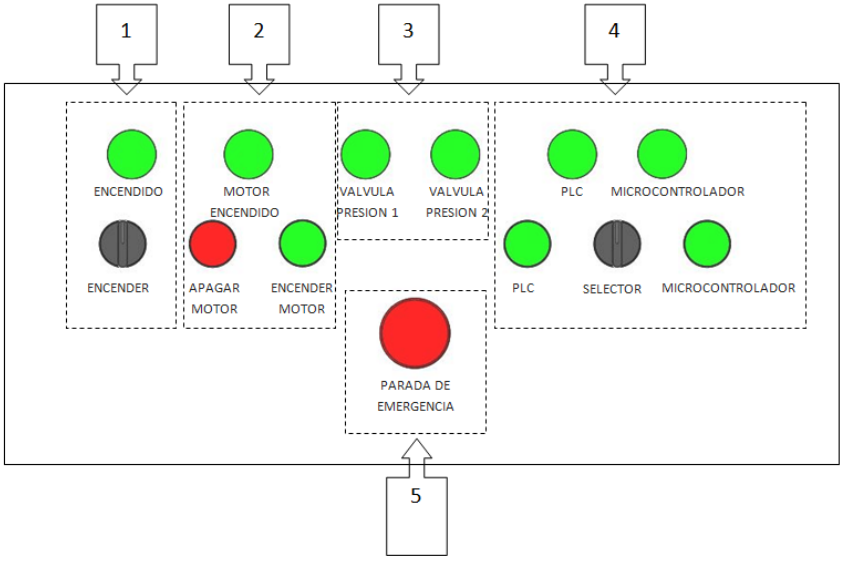
- Zona 2. Contiene los pulsadores para el encendido y apagado del motor principal así como también el indicador de motor encendido.

- Zona 3. Contiene los pilotos indicadores del estado de presión

- Zona 4. Contiene el selector de energía para el PLC o Microcontrolador, sus respectivos pilotos, pulsadores de Reset para el programa.

- Zona 5. Contiene la parada de emergencia

Figura 67. Esquema de zonas del tablero de mando.

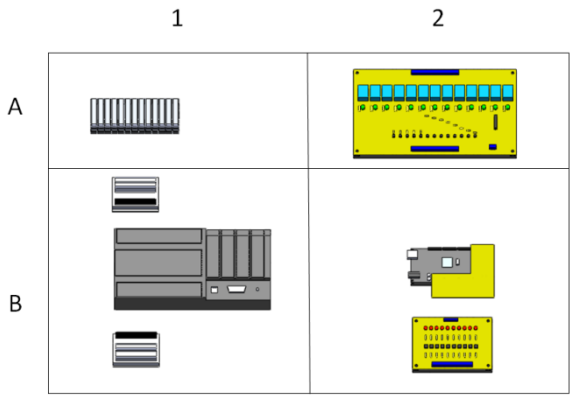


Fuente: Autores.

4. DISEÑO DE LAS ZONAS DE TRABAJO

Los dispositivos de control fueron ubicados en el panel de soporte, de tal forma que sea fácil su apreciación y manipulación. El panel se dividió en cuatro zonas como se puede apreciar en la figura 68.

Figura 68. Zona en el panel de montaje.



Fuente: Autores.

4.1 ZONA A1. Se encuentran doce bornes portafusibles con una capacidad de 0.5amp, estos se utilizan para proteger las salidas del PLC.

4.2 ZONA A2. Se encuentra la tarjeta de potencia, la cual recibe las señales procedentes del PLC y el Microcontrolador, para amplificarlas y enviarlas a los múltiples dispositivos a controlar.

4.3 ZONA B1. Se encuentra el PLC, así como también dos bloques de Bornes, los cuales permiten conectar el PLC con las señales de entrada y salida, estos se encuentran abajo y arriba respectivamente.

4.4 ZONA B2. Se encuentra el Microntrolador y la tarjeta de interface de señales de entrada.

5. DESCRIPCIÓN DEL SISTEMA ELECTROHIDRÁULICO A CONTROLAR

La figura 69 muestra los elementos del banco de prensas y malacate el cual cuenta con un paquete hidráulico de potencia conformado por un motor trifásico de 14 HP (11 KW) y una bomba de paletas Vickers que entrega un caudal máximo de 8.75 GPM y una presión máxima, limitada por la potencia del motor eléctrico y la válvula de seguridad del banco, de 120 Bar (1740psi). Posee un segundo nivel de seguridad a 50 Bar (750 Psi) que puede ser utilizado en cualquier instante para la realización de las prácticas, haciendo las conexiones eléctricas necesarias[12]. El esquema hidráulico se puede observar en la figura 70.

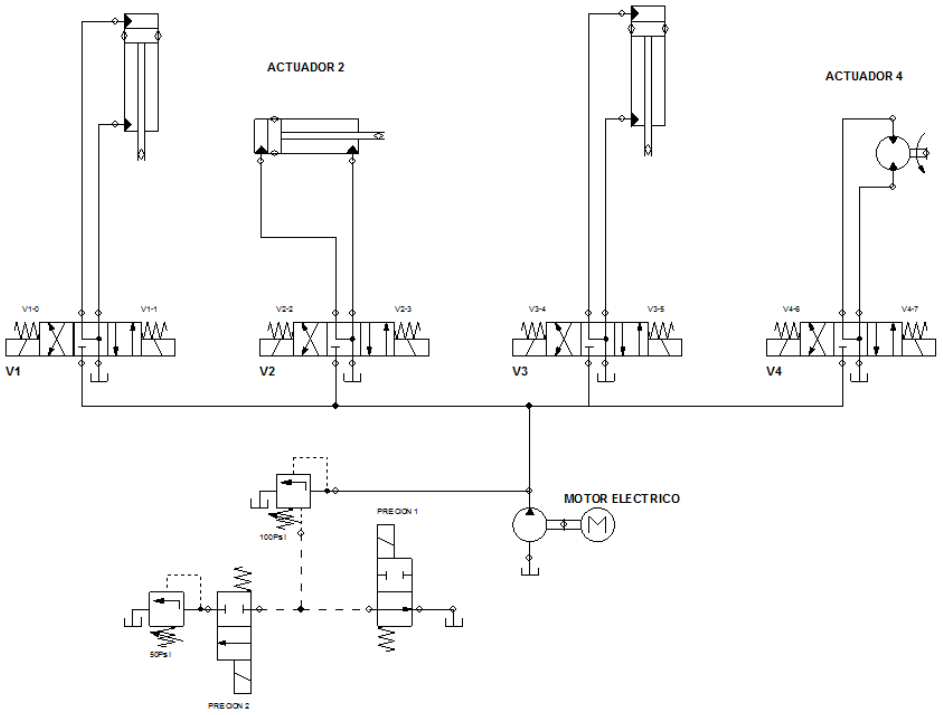
El banco contiene una sección de relés electromagnéticos y temporizadores eléctricos dispuestos de tal forma para hacer prácticas de armado de cableados para elaborar lógicas eléctricas con el fin de controlar los circuitos hidráulicos. Es por esta razón que las electroválvulas no están conectadas directamente, solamente se cuenta con los terminales de conexión de los solenoides dispuestos en el frente del banco para su conexión de forma didáctica.

Figura 69. Elementos del banco de prensa y malacate.

Fuente: Autores.

[12]Fuente: Florez Saúl; diseño, elaboración e implementación de prácticas para el laboratorio de Potencia Fluida. Bucaramanga 2004. Trabajo de grado (Ingeniería mecánica) universidad Industrial de Santander. Escuela de Ingeniería mecánica.

Figura 70. Esquema hidráulico a controlar.



Fuente: Autores.

Como actuadores posee tres cilindros de doble efecto y un motor hidráulico de paletas (ver figura 71), los cuales son controlados por 4 electroválvulas direccionales 4/3 con centros tipo “punto flotante” en circuitos hidráulicos independientes.

Figura 71. Motor eléctrico y bomba hidráulica.

Fuente: Autores.

La figura 72 muestra el actuador uno, el cual se encuentra ubicado en forma vertical y actúa contra resortes para lograr el efecto de prensas hidráulicas, este actuador es accionado por la electroválvula Vickers que se ilustra en la figura 73.

Figura 72. Actuador 1.

Fuente: Autores.

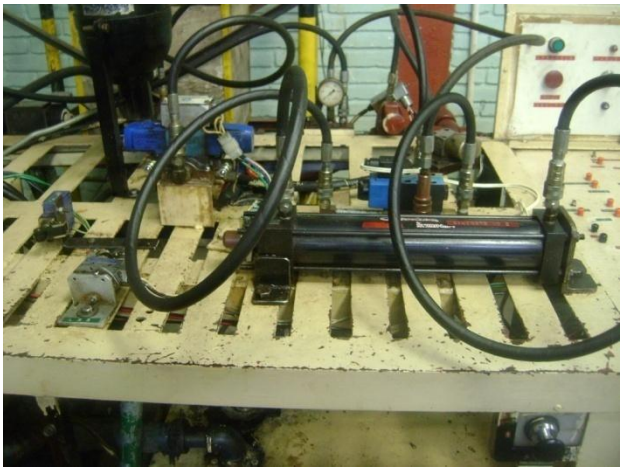
Figura 73. Electroválvula 1.

Fuente: Autores.

La figura 74 ilustra el actuador dos, el cual se encuentra ubicado de forma horizontal sobre el banco. De igual manera que los actuadores uno y dos este es

controlado por una electroválvula Vickers la cual se puede observar en la figura 75.

Figura 74. Actuador 2.



Fuente: Autores.

Figura 75. Electroválvula 2.

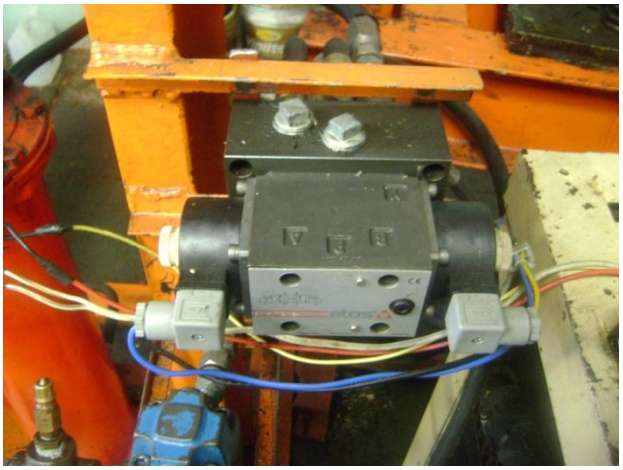
Fuente: Autores.

La figura 76 muestra el actuador tres el cual también está ubicado de forma vertical e igualmente que el actuador 1 presiona unos resortes, para lograr el efecto de circuitos de prensas hidráulicas, la figura 77 muestra la electroválvula que controla este actuador.

Figura 76. Actuador 3.

Fuente: Autores.

Figura 77. Electroválvula 3.



Fuente: Autores.

Como elementos para simular carga se cuenta un elevador provisto con un peso de 500 Kg que se puede observar en la figura 78, el cual es accionado por el motor hidráulico (ver figura 79), a través de un reductor planetario de velocidad,

además de una válvula sobrecentro doble que controla el ascenso y descenso de la carga, la cual se ilustra en la figura 80.

Figura 78. Elevador de carga.

Fuente: Autores.

Figura 79. Motor hidráulico.

Fuente: Autores.

Figura 80. Electroválvula 4.

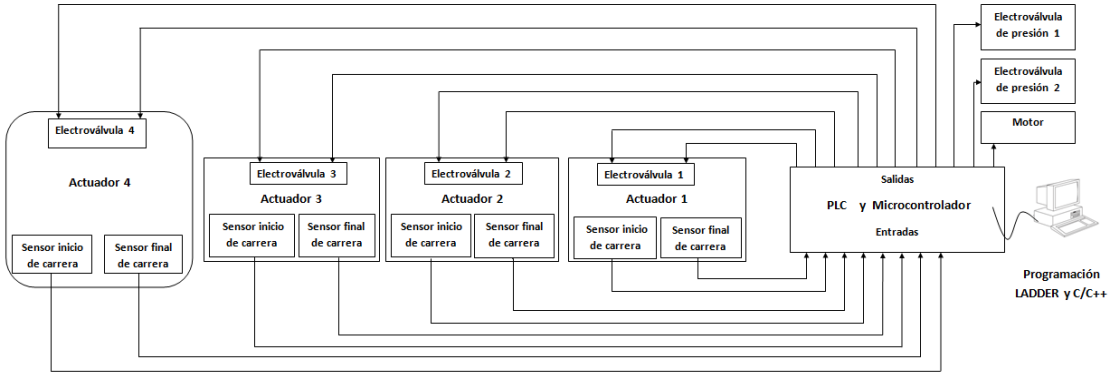
Fuente: Autores.

6. DISEÑO DEL SISTEMA ELECTRICO

6.1ESQUEMAS GENERALES.

La figura 81 muestra un esquema general del sistema de control, en ella se pueden apreciar los cuatro actuadores a controlar con sus respectivos sensores y electroválvulas.

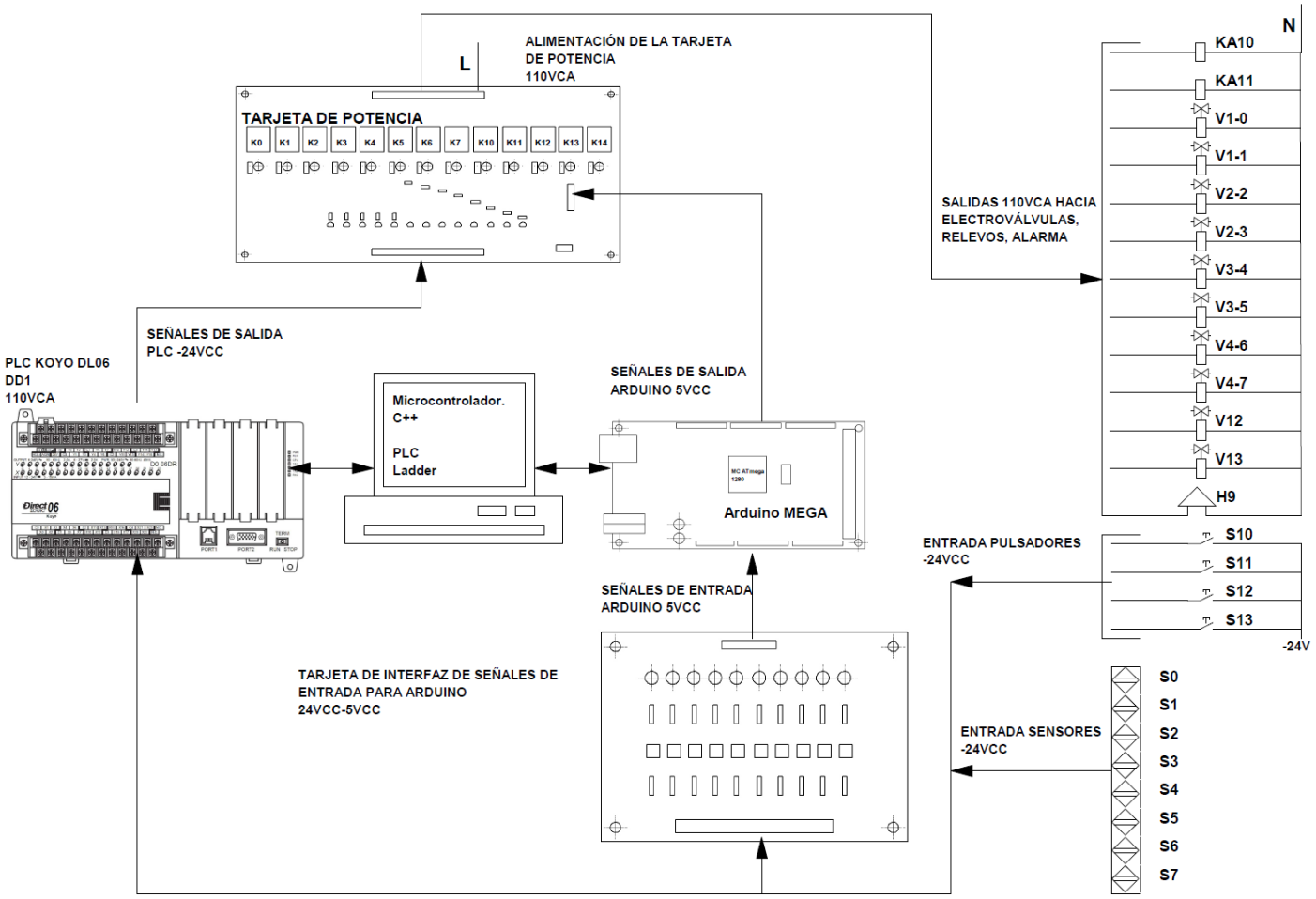
Figura 81. Esquema general del sistema de control.



Fuente: Autores.

En la figura 82, se muestra en esquema básico del circuito eléctrico del sistema de control, que está conformado básicamente por, un PLC, un Microntrolador, una tarjeta de interface de señales de entrada, y una tarjeta de potencia.

Figura 82. Esquema básico del sistema de control.



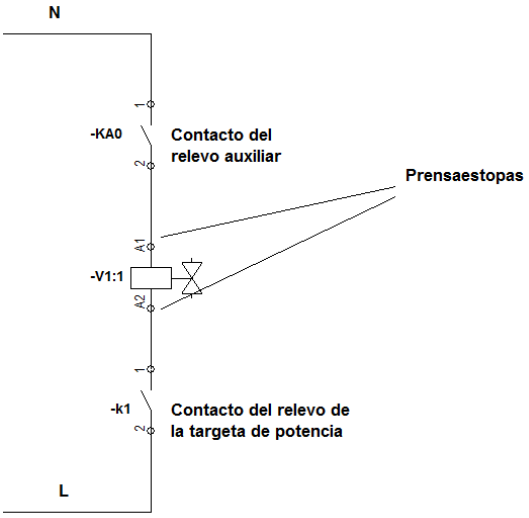
Fuente: Autores.

6.2 CIRCUITO ELÉCTRICO DE LAS ELECTROVÁLVULAS.

El banco hidráulico actualmente es controlado por relevadores y temporizadores, los cuales energizan las electroválvulas que a su vez accionan los actuadores hidráulicos, la energía que proveniente de los temporizadores y relevos es cableada por los estudiantes mediante prensaestopas, las líneas de 120 VAC y N que llegan a estas prensaestopas se pueden cablear sin importar el polo que se escoja de las electroválvula; A diferencia del sistema de control a instalar, estos polos deben estar fijos, de esta manera se puede lograr una fácil conexión y desconexión del sistema de control con las electroválvulas y los sensores ubicados en el banco hidráulico. Como los dos sistemas deben funcionar de manera paralela, fué necesario aislar los polos de lado neutro de las electroválvulas para evitar cortos circuito en el momento que los estudiantes estén cableando.

La Fase de 110VAC está aislada por los relevos que están ubicados en la tarjeta de potencia, mediantes contactos que permanecen normalmente abiertos cuando los relevos no son accionados, mientras que la fase N se aisló mediante relevos electromagnéticos situados en el tablero de potencia, estos se activan automáticamente cuando se acciona el selector que energiza el banco del sistema de control, esto se puede apreciar en la figura 83.

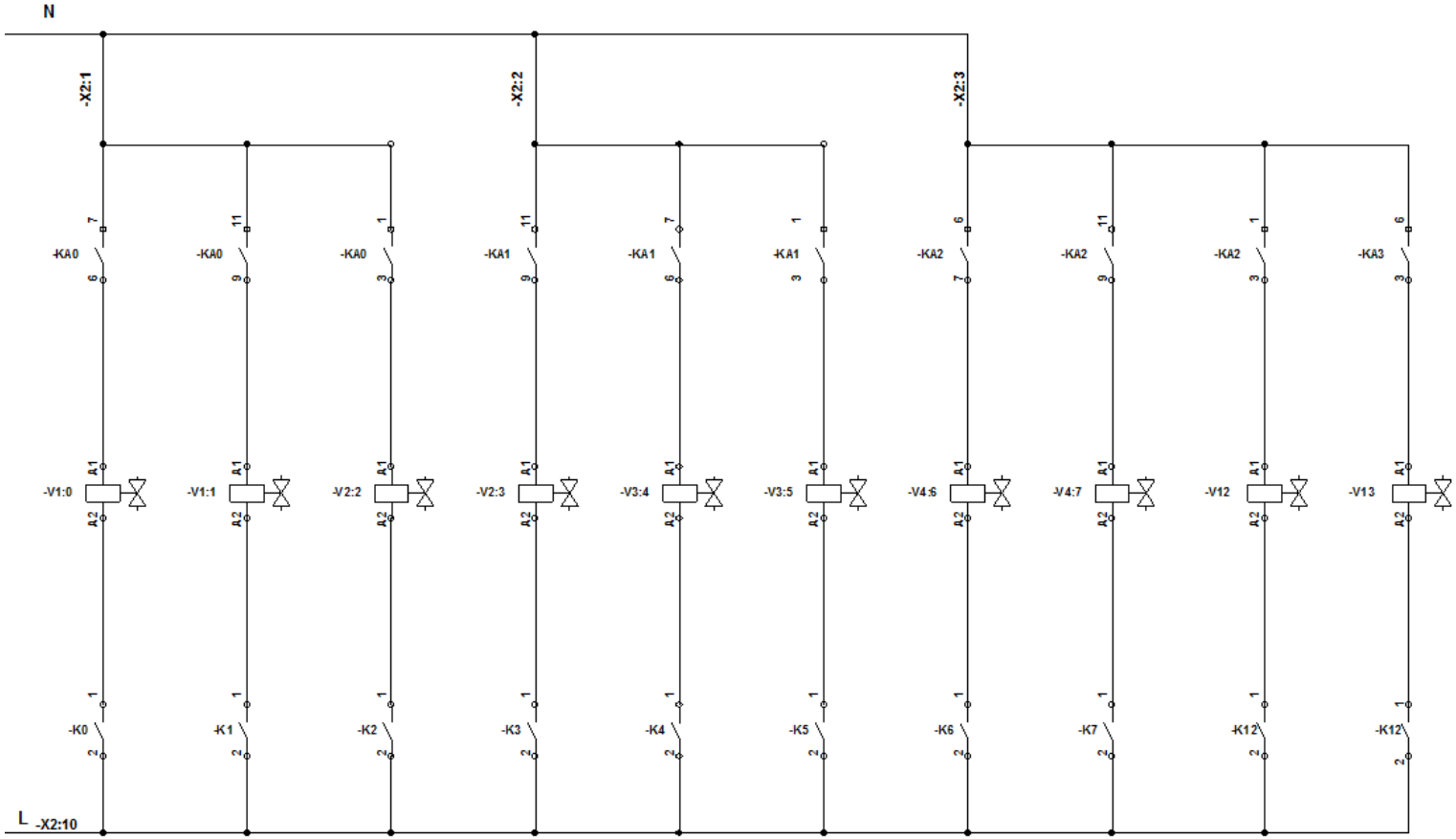
Figura 83. Esquema básico del aislamiento una electroválvula.



Fuente: Autores.

En la figura 84 se observa el esquema eléctrico de las electroválvulas, en el se pueden observar los contactos auxiliares KA para la aislación de la línea neutro, y los contactos K de los relevos ubicados en la tarjeta de potencia, los cuales son activados por las señales provenientes del PLC y del Microcontrolador.

Figura 84. Esquema eléctrico de las electroválvulas

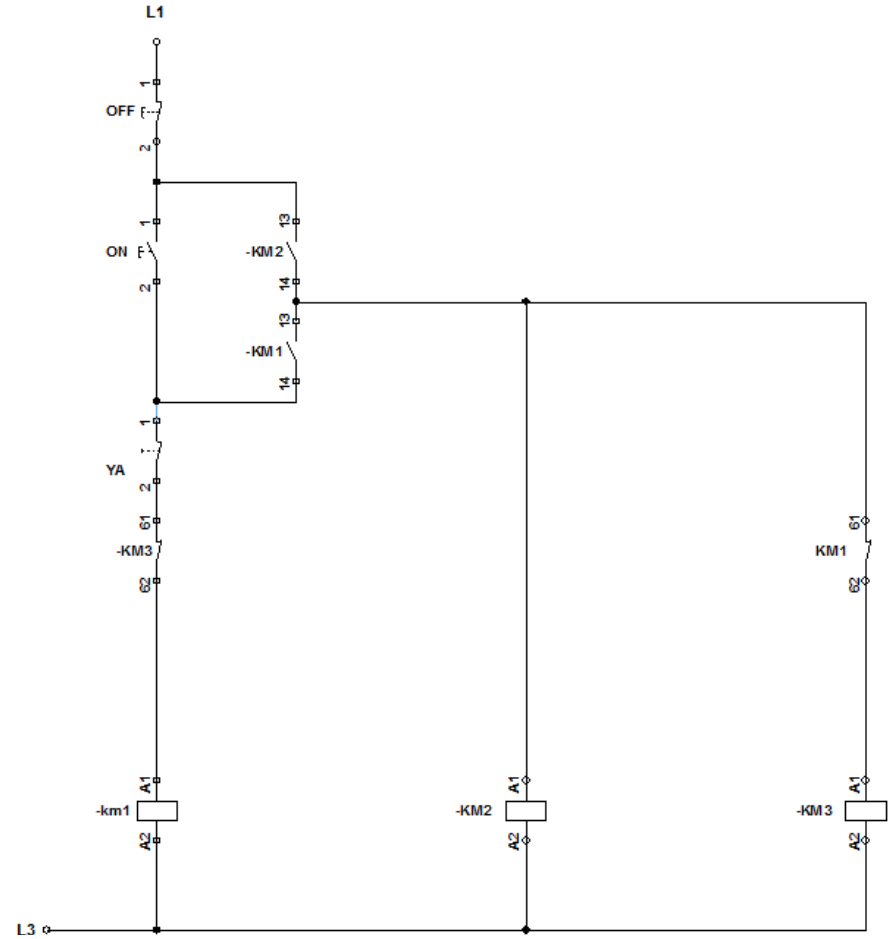


Fuente: Autores

6.3 CIRCUITO ELÉCTRICO DEL ARRANQUE DEL MOTOR.

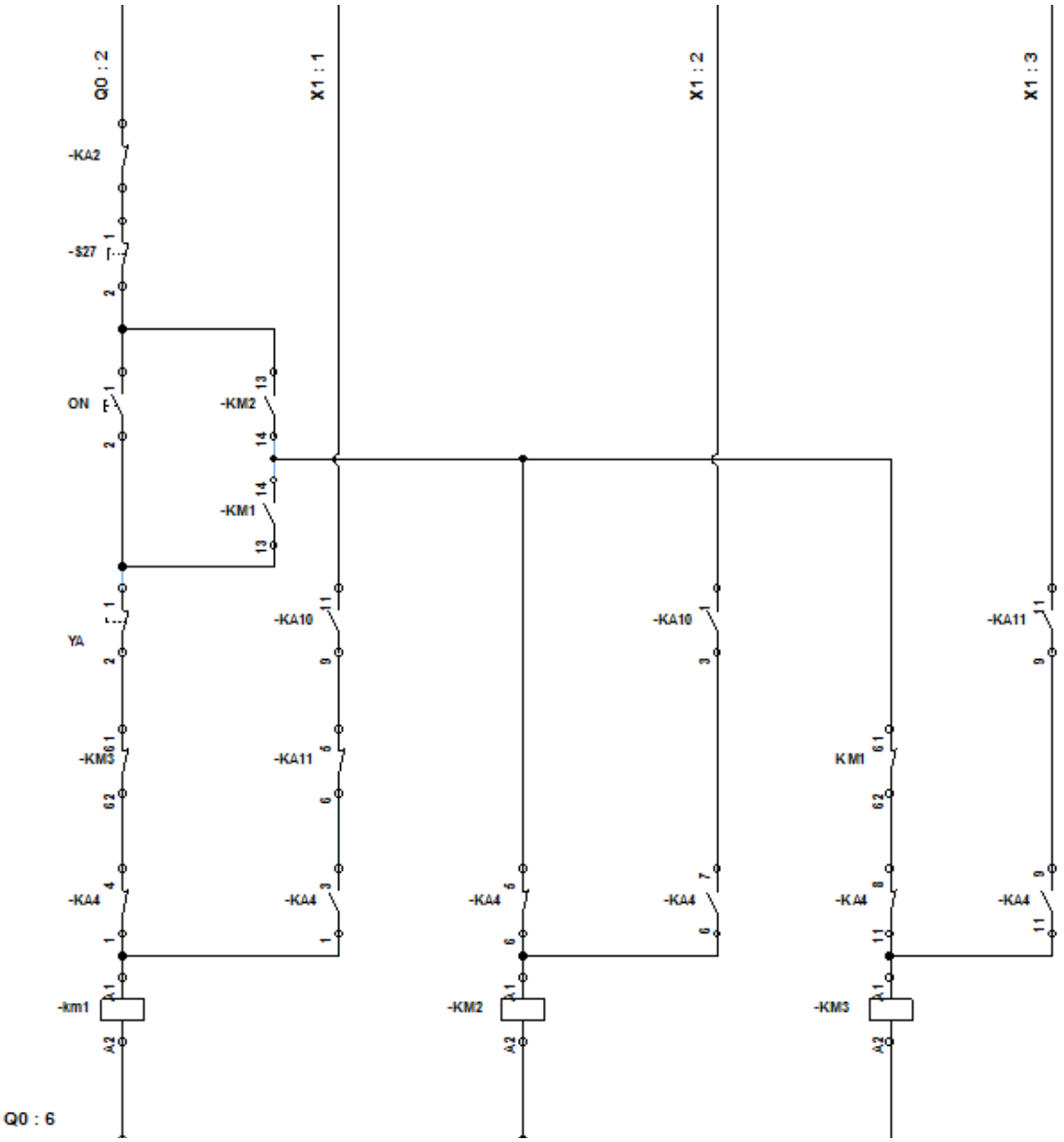
El encendido del motor también será gobernado por el sistema de control, por lo tanto fué necesario diseñar un circuito eléctrico que permitiera el arranque de este por cualquiera de los dos sistemas. En la figura 85 se observa el esquema eléctrico antiguo de los contactores para el arranque del motor. Para adaptar el circuito al nuevo sistema de control las líneas de activación que energizan las bobinas KM1, KM2, KM3 de los arrancadores del motor, fueron acopladas a contactos normalmente cerrados de relevos situados en el tablero de potencia, estos contactos se abren automáticamente en el momento de accionar el selector que energiza el sistema de control para dar paso al el nuevo sistema eléctrico que gobernará el arranque del motor. Las nuevas líneas de accionamiento se encuentran acopladas a contactos normalmente abiertos del mismo relevo de tal manera que al activarse el modo automático estos se cerraran. La figura 86 muestra el sistema eléctrico con las modificaciones necesarias para que el arranque del motor sea pueda realizar por cualquiera de los dos sistemas de control.

Figura 85. Antiguo esquema eléctrico de los contactores para el arranque del motor.



Fuente: Autores.

Figura 86.Esquema eléctrico de los contactores para el arranque del motor.



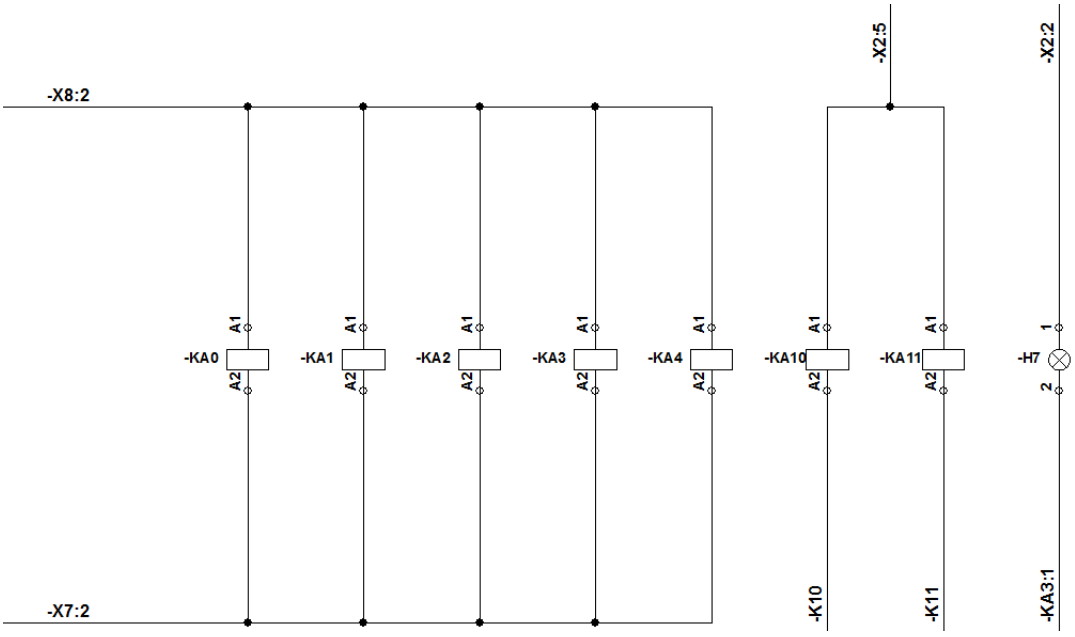
Fuente: Autores.

6.4. CIRCUITO ELÉCTRICO DEL TABLERO DE POTENCIA.

En el tablero de potencia se encuentran ubicados, los relevos de aislamiento para las electroválvulas, los relevos de aislamiento para el sistema eléctrico del arranque del motor, los relevos que encienden el motor mediante el sistema de control.

La figura 87 muestra el circuito eléctrico de los relevos auxiliares, los relevos KA0, KA1, KA2, KA3, KA4 se activan por medio de una señal que proviene del banco de didáctico, la cual se envía al momento de mover el selector principal del banco al estado encendido. Los relevos KA10 y KA11 son activados por las señales provenientes de la tarjeta de potencia.

Figura 87. Circuito eléctrico del accionamiento de los relevos auxiliares.

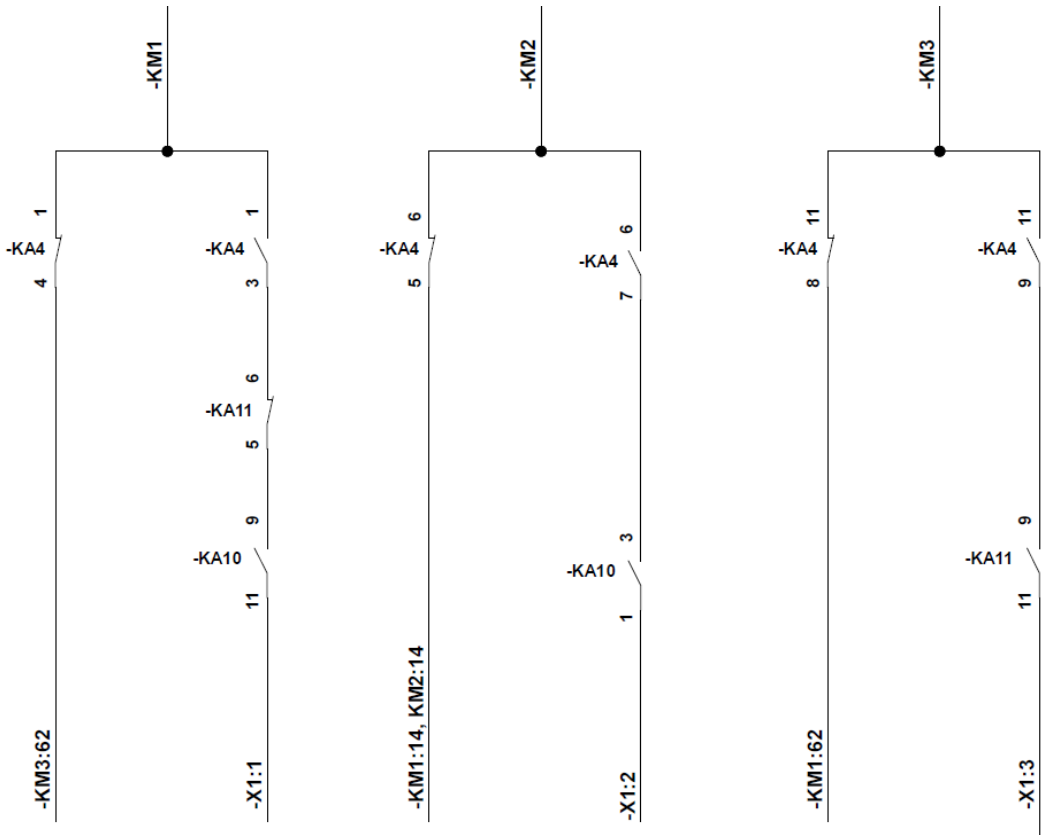


Fuente: Autores.

La figura 88 muestra los contactos del relevo auxiliar KA4, que se utilizan para aislar el sistema antiguo y dar paso al nuevo sistema de control, también se

ilustran los relevos KA10 Y KA11 los cuales respectivamente encienden el motor en modo estrella y luego triangulo.

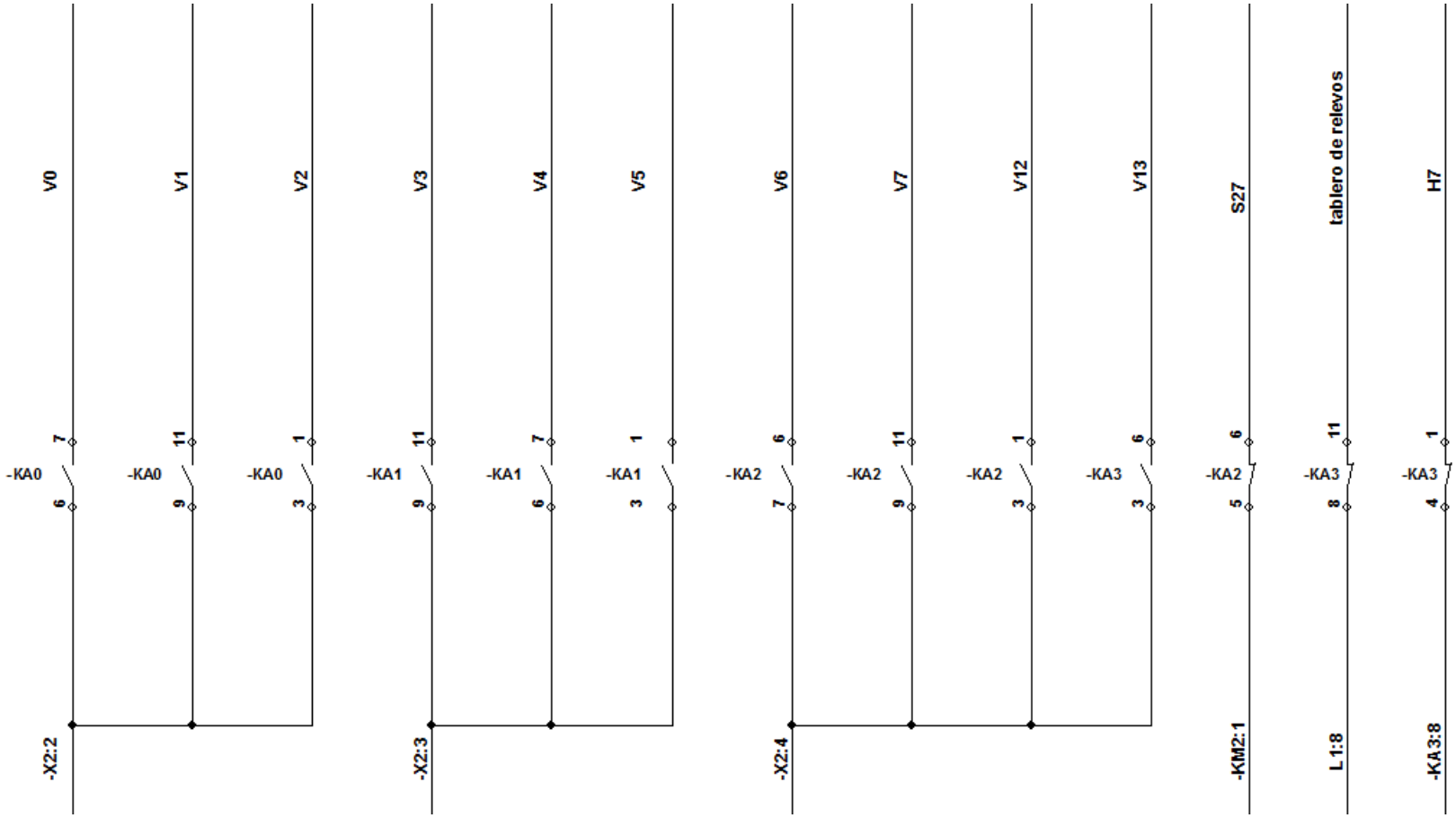
Figura 88. Circuito eléctrico de los contactos para el accionamiento del motor.



Fuente: Autores.

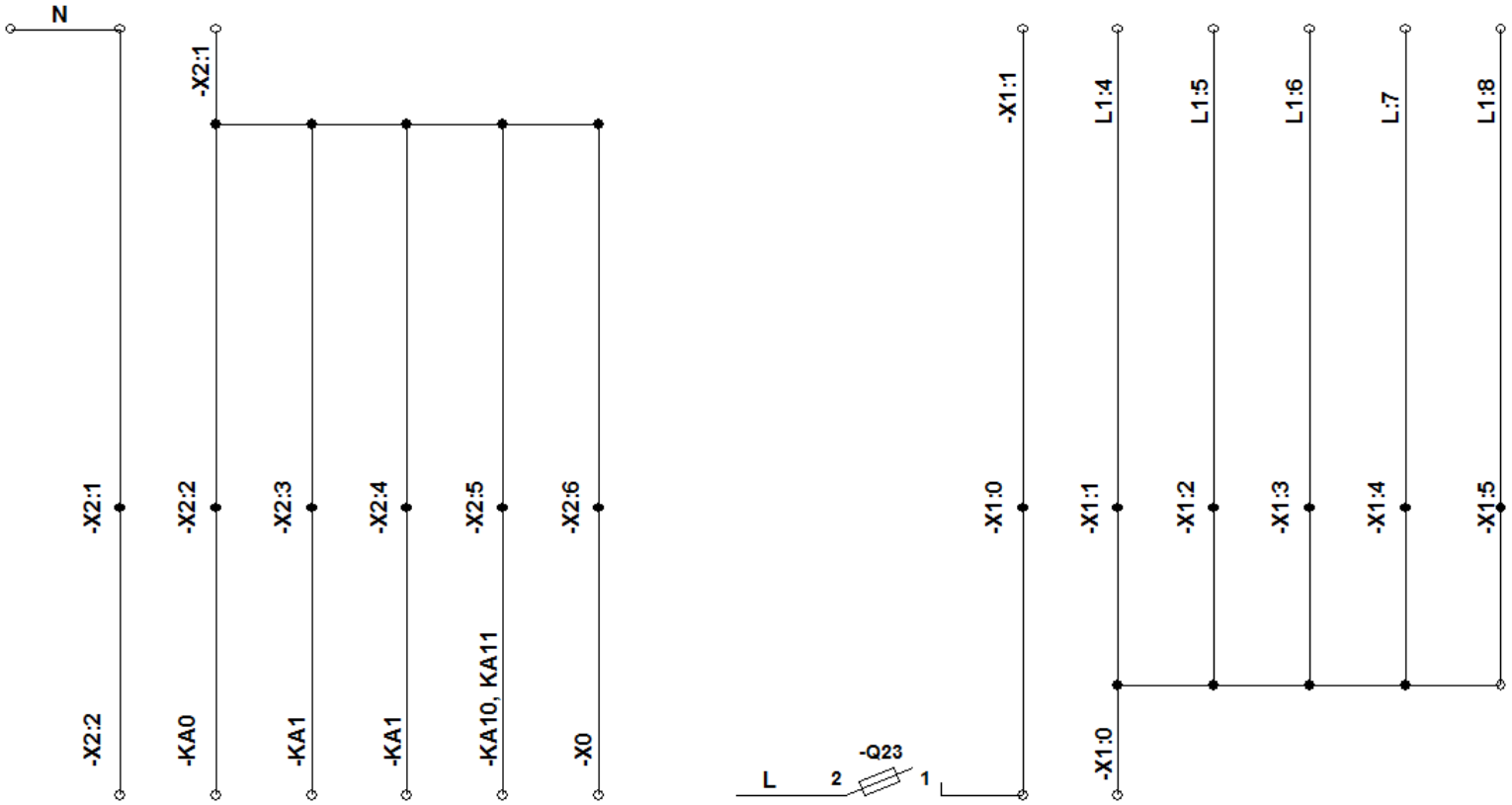
La figura 89 ilustra el circuito eléctrico de los contactos auxiliares de los relevos KA0, KA1, KA2 y KA3 que aíslan las líneas neutro de las electroválvulas, y en la figura 90 se ilustran las conexiones de los bornes en el tablero de potencia.

Figura 89. Circuito eléctrico de los contactos auxiliares para la aislación de la línea neutro de las electroválvulas.



Fuente: Autores.

Figura 90. Bornes del tablero de potencia.



Fuente: Autores.

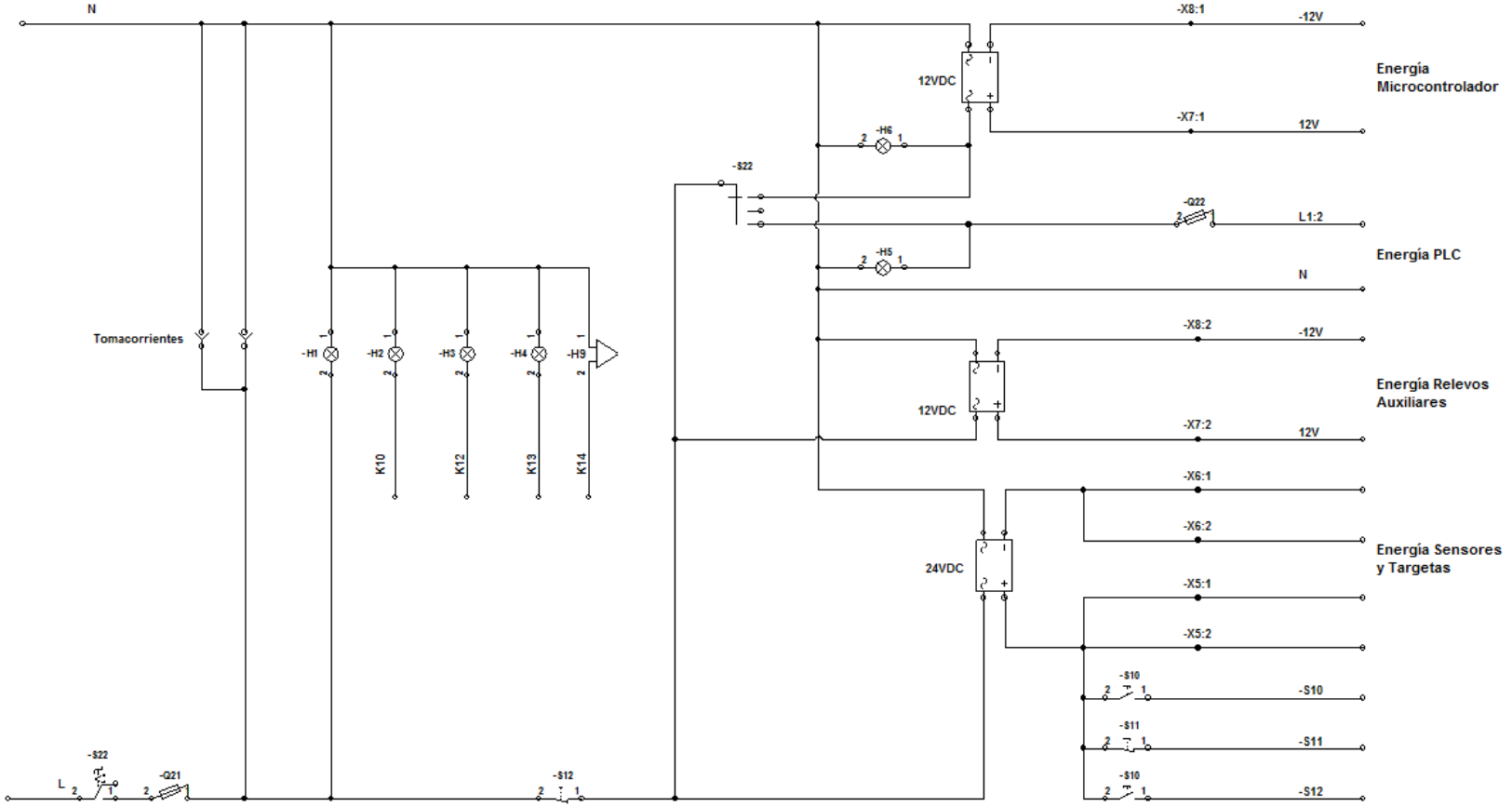
6.5 TABLERO ELÉCTRICO

En este se instaló el sistema eléctrico del banco de experimentación. Las señales que provienen de los sensores se conectan al tablero eléctrico por medio de un puerto paralelo LTP de veinticinco pines, las señales de salidas también se comunican al exterior por medio de puertos LTP, de igual manera lo hacen las señales auxiliares. El circuito eléctrico principal está protegido un fusible de acción rápida CHINT C16 referenciado en la figura 91 como -Q21. El PLC tiene un fusible de acción rápida independiente Schneider C1 lo cual garantiza mayor protección para el mismo.

Las tensiones de corriente continua de 24 y 12 voltios requeridas para el funcionamiento de algunos componentes se obtienen de transformadores de corriente. Se usan tres en total dos de 110 VAC a 12 VDC, uno para la energía principal de la tarjeta de control Arduino y otro para la activación de los relevos auxiliares. El otro transformador es de 110 VAC a 24 VDC, que se utiliza para energizar los sensores y también para algunos componentes como tarjetas de potencia, entradas y salidas del PLC, tarjeta de interface de señales de entra.

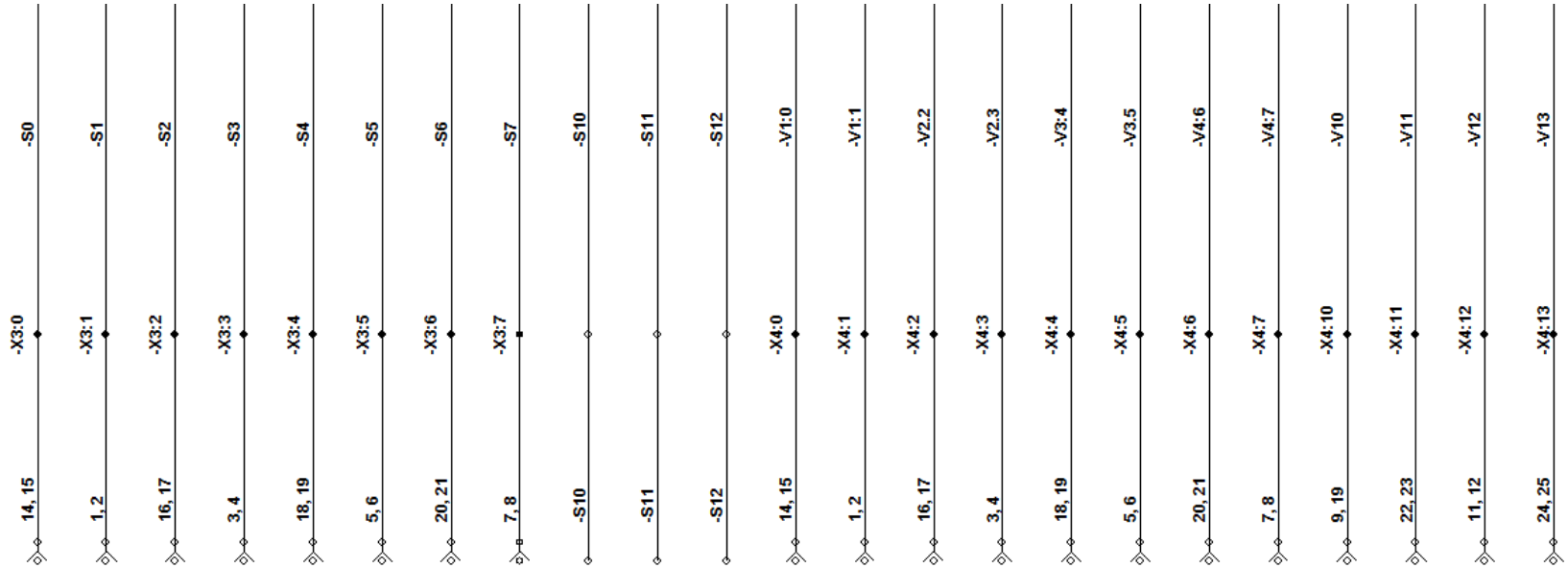
En la figura 91 se ilustra parte del esquema eléctrico del tablero eléctrico. La figura 92 ilustra el esquema eléctrico de los bornes de las señales y la figura 93 ilustra el esquema eléctrico de los bornes de tensiones.

Figura 91. Esquema electrico del tablero electrico.



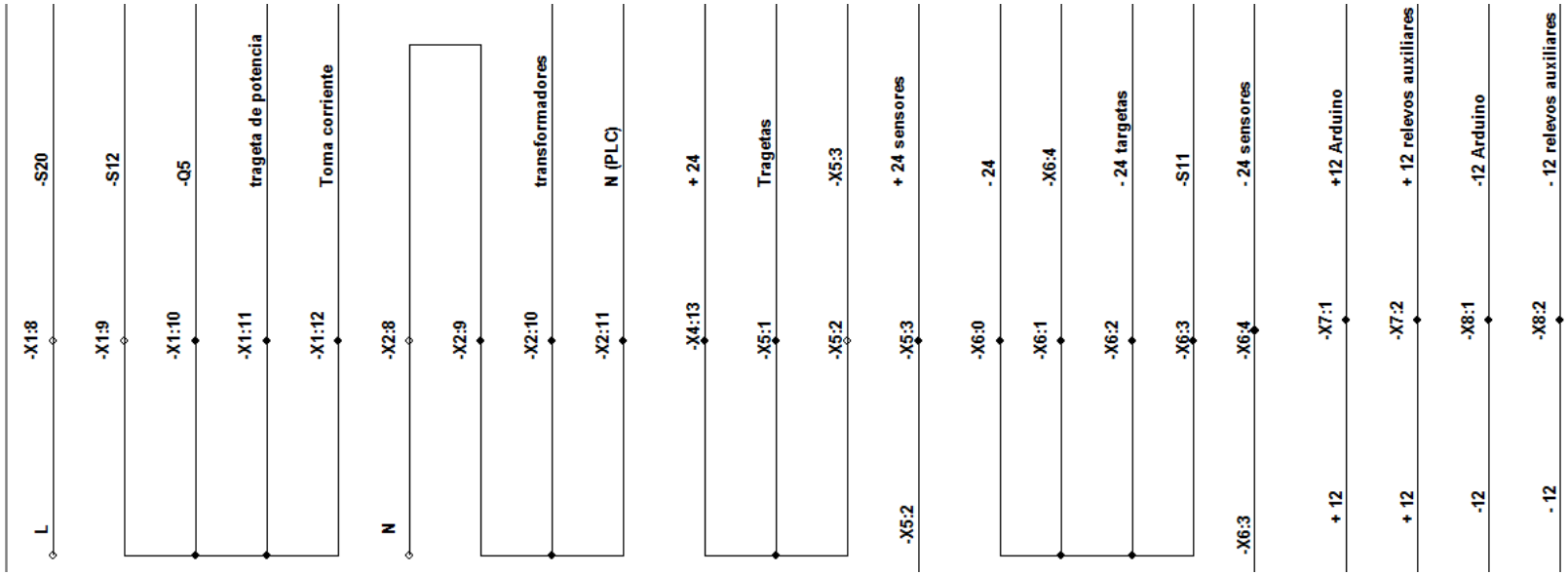
Fuente: Autores.

Figura 92. Esquema eléctrico de los bornes de las señales.



Fuente: Autores.

Figura 93. Esquema eléctrico de los bornes de tensiones.



Fuente: Autores.

7. CONSTRUCCIÓN DEL BANCO DE EXPERIMENTACIÓN

7.1 INSTALACIÓN DE LOS SENSORES

Se instalaron ocho sensores al inicio y final de carrera de cada actuador, por medio de perfiles en lámina de acero, atornilladas a la estructuras. El cableado va cubierto por un tubo flexible ranurado de plástico para su protección. A continuación se puede observar la distribución de los sensores en el banco hidráulico.

7.1.1 Sensor 0. Se encuentra ubicado en el inicio de carrera del actuador uno, tal como se aprecia en la figura 94.

Figura 94. Sensor 0.

Fuente: Autores.

7.1.2 Sensor 1. Se encuentra ubicado al final de carrera del actuador uno, tal como se aprecia en la figura 95.

Figura 95. Sensor 1.

Fuente: Autores.

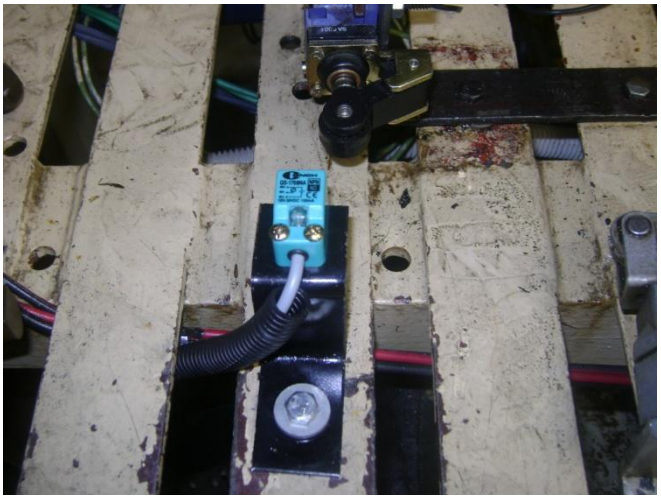
7.1.3 Sensor 2. Se encuentra ubicado en el inicio de carrera del actuador dos, tal como se aprecia en la figura 96.

Figura 96. Sensor 2.

Fuente: Autores.

7.1.4 Sensor 3. Se encuentra ubicado al final de carrera del actuador dos, tal como se aprecia en la figura 97.

Figura 97. Sensor 3.



Fuente: Autores.

7.1.5 Sensor 4. Se encuentra ubicado en el inicio de carrera del actuador tres, tal como se aprecia en la figura 98.

Figura 98. Sensor 4.

Fuente: Autores.

7.1.6 Sensor 5. Se encuentra ubicado al final de carrera del actuador tres, tal como se aprecia en la figura 99.

Figura 99. Sensor 5.

Fuente: Autores.

7.1.7 Sensor 6. Se encuentra ubicado la parte inferior de la carga, tal como se aprecia en la figura 100

Figura 100. Sensor 6.

Fuente: Autores.

7.1.8 Sensor 7. Se encuentra ubicado la parte superior de la carga, tal como se aprecia en la figura 101.

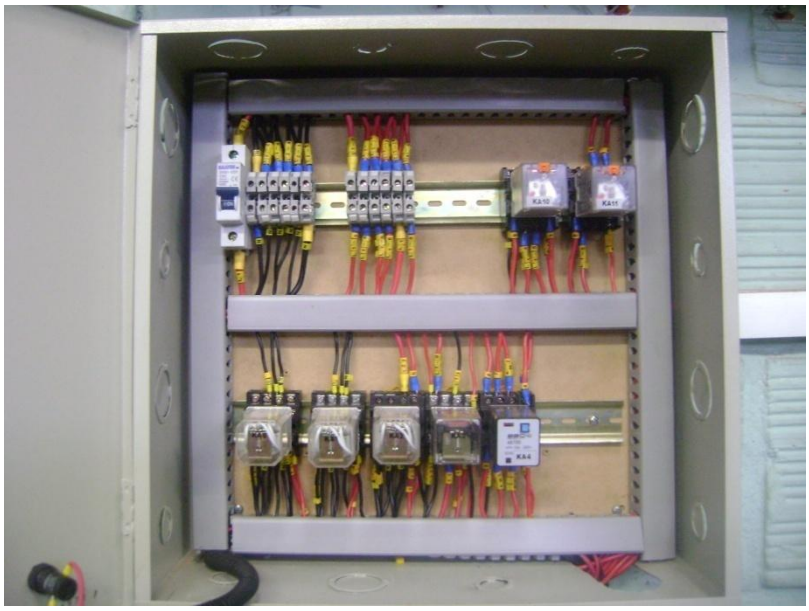
Figura 101. Sensor 7.

Fuente: Autores.

7.2 CONSTRUCCIÓN DEL TABLERO DE POTENCIA

Tiene unas dimensiones de 40cm x 40cm, los componentes se instalan sobre rieles Din y estos a su vez se aseguran sobre una placa de madera, además contiene canaletas de 25cm x 25cm para el alojamiento del cableado. En el tablero se instalaron cinco relevos auxiliares accionados a 12 VDC, para el aislamiento de la línea neutro de las electroválvulas y de las fase L1 requerida para el arranque del motor eléctrico, dos relevos auxiliares accionados a 110AC que energizan las boninas de los contactores que encienden el motor principal. Está provisto por un piloto que se enciende al momento de energizar los relevos auxiliares indicando su activación. Mediante canaletas eléctricas se transmiten los cables hacia el banco hidráulico y hacia el sistema de control. El tablero de potencia se puede apreciar en la figura 102.

Figura 102. Tablero de potencia.



Fuente: Autores.

7.3 CONSTRUCCIÓN DE LA ESTRUCTURA SOPORTE DEL SISTEMA DE CONTROL.

7.3.1 Estructura principal. Una vez definido el diseño de la estructura de soporte para el sistema de control, se procedió a su construcción. El armazón principal que se observa en la figura 103, se construyó con laminas de acero Cold Rollet número 18 y 16, y con tubería Cold Rollet de 1.5 pulgadas calibre 18, está dotado de una mesa de madera en MDF encapada en fórmica, que sirve como espacio de trabajo. La estructura está provista por ruedas con seguro que permiten su fácil desplazamiento al mismo tiempo que ofrece seguridad al momento de la elaboración de las prácticas.

La estructura principal consta de cuatro gavetas que tienen unas dimensiones de 20 cm de alto por 35 cm de ancho y 45 cm de profundidad, estas brindan el espacio suficiente para guardar los diferentes accesorios requeridos para el funcionamiento del banco. En la figura 104 se puede observar las principales partes que constituyen la estructura.

Figura 103. Estructura principal.

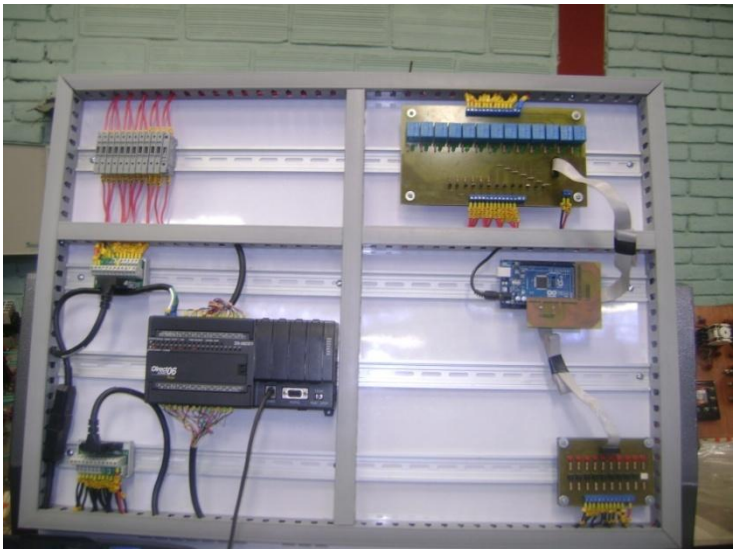
Fuente: Autores.

Figura 104. Despiece de la estructura.

Fuente: Autores.

7.3.2 Panel para soporte de los componentes de control. Está hecho en lámina de acero Cold Rollet número 18, soportadas sobre barras de acero de 1 pulgada número 18 del mismo material, el panel está recubierto por una lámina de MDF enchapada en fórmica atornillada al panel, además está provista por cuatro rieles Din en aluminio y canaletas ranuradas de 25x40 mm. Un tubo ranurado plástico conduce el cableado del tablero eléctrico hasta el panel, llegando a este por la parte trasera. La figura 105 ilustra el panel con los componentes de control instalados

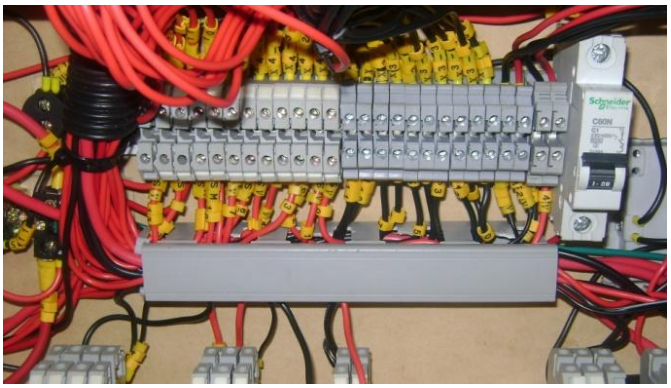
Figura 105. Panel con los componenetes instalados.



Fuente: Autores.

7.3.3 Tablero eléctrico. Aloja todo el sistema eléctrico del banco didáctico. La figura 106 ilustra la sección de bornes de las señales de entradas, salidas, y auxiliares.

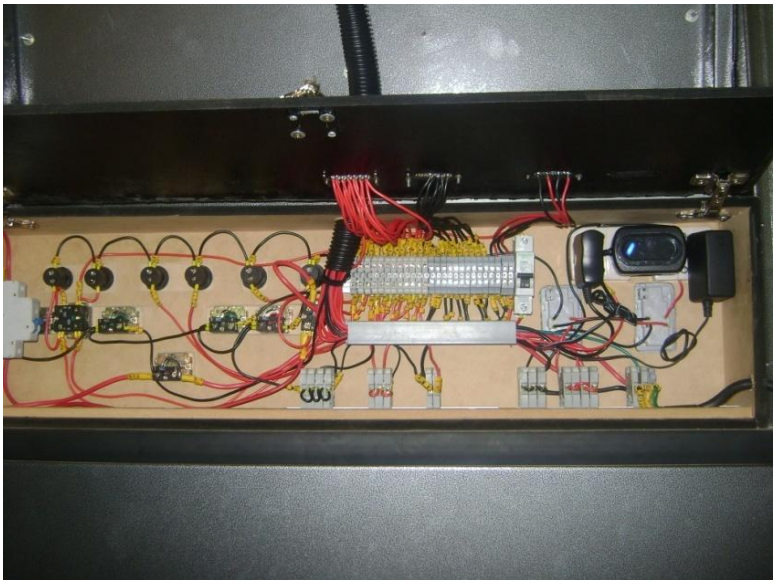
Figura 106. Bornes de las señales.



Fuente: Autores.

En la figura 107 se observa todo el sistema de cableado instalado en el tablero eléctrico, incluyendo el tablero de mando.

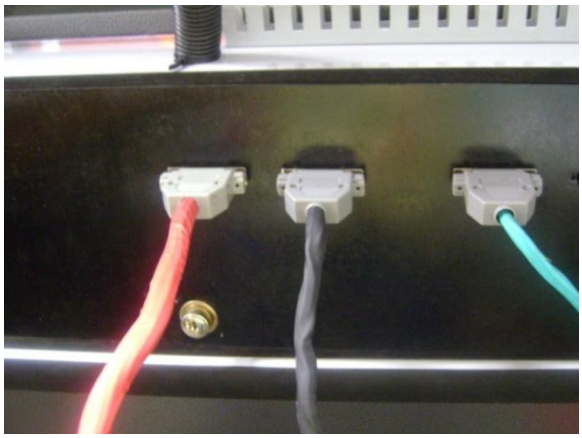
Figura 107. Tablero eléctrico.



Fuente: Autores.

El tablero eléctrico cuenta con tres conectores tipo impresora (ver Figura 108) por medio de los cuales llegan las señales provenientes de los sensores, salen las señales del sistema de control hacia los distintos dispositivos a controlar, y se proporcionan las tensiones requeridas por los relevos auxiliares y sensores.

Figura 108. Conexión tipo impresora para las señales.



Fuente: Autores.

7.3.4 Tablero de mando. Se encuentra ubicado en la parte frontal del tablero eléctrico, en un lugar de fácil visualización y acceso (Ver Figura 109).

Figura 109. Tablero de mando.

Fuente: Autores.

La Figura 110 ilustra el banco didáctico con todos los componentes de control instalados sobre este.

Figura 110. Banco didáctico.

Fuente: Autores.

8. DISEÑO DEL MANUAL DE PRÁCTICAS

Por medio de este manual se brindaran las herramientas necesarias para realización de las diferentes prácticas por parte de los estudiantes, además servirá como soporte para el profesor y auxiliar del laboratorio en la ejecución de las mismas y de esta manera valorar el desempeño de los estudiantes.

El manual de prácticas está diseñado de tal manera que se puedan aprovechar al máximo los recursos con los que se cuentan para la elaboración de las diferentes actividades programadas. Este manual tiene la siguiente estructura.

8.1 PROCEDIMIENTOS DE SEGURIDAD.

Se exponen las diferentes medidas de seguridad que se deben cumplir para una adecuada realización de las prácticas, minimizando de esta manera el riesgo. Algunas de estas son las siguientes:

- Ubique la parada de emergencia antes de comenzar las prácticas.
- Antes de comenzar la práctica asegúrese de retirar los cables en el tablero de control de lógica cableada.
- Siga estrictamente el ítem de instalación del cableado.
- Asegúrese de programar el segundo nivel de presión en el momento de bajar la carga que está acoplada al actuador 4 (Motor).
- La operación de los equipos debe ser supervisada por el auxiliar del laboratorio.
- El área de trabajo donde realizaran las prácticas debe estar libre de objetos ajenos al material y equipo de trabajo.
- No como ni beba en el laboratorio.
- Está prohibido fumar en el laboratorio.

8.2 INSTALACIÓN DEL CABLEADO.

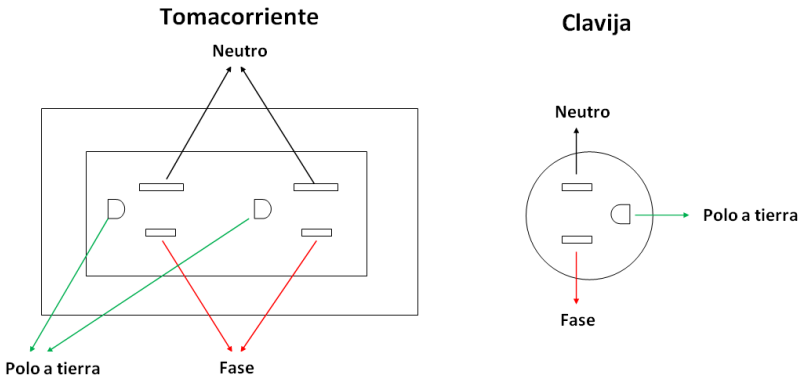
Se encuentran las indicaciones pertinentes instalar el cableado que se requiere para la elaboración de las prácticas.

8.3 ALIMENTACIÓN PRINCIPAL DEL BANCO DE EXPERIMENTACIÓN.

Asegúrese de conectar el banco de experimentación a una instalación eléctrica de 110-120 VAC que cumpla con los estándares eléctricos.

Debido a las instalaciones internas del banco de experimentación, se debe asegurar que la línea de fase de la clavija coincida con el tomacorriente. Si se conecta a un tomacorriente no polarizado asegúrese de conectar correctamente la línea de fase tal como se muestra en la figura 111.

Figura 111. Conexiones para la toma principal de energía del banco.



Fuente: Autores.

8.4 CABLEADO DE SEÑALES.

Asegúrese de conectar correctamente los cables de señales en sus respectivos puertos. Observe el color de cada uno de ellos en la figura 112.

Figura 112. Colores de los cables de señales del banco.

Fuente: Autores.

8.4.1 Instalación del PLC.

- Conecte la energía principal del PLC, como se muestra en la Figura 113.
- Conecte el cable de señales de entrada al bloque de bornes inferior. como se muestra en la figura 113.
- Conecte el cable de señales de salida al bloque de bornes inferior. como se muestra en la figura 113.

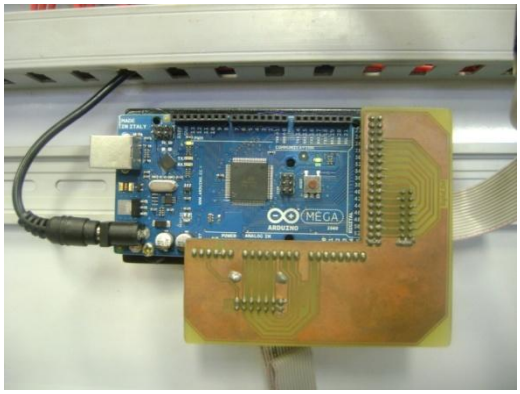
Figura 113. Conexiones del PLC.

Fuente: Autores.

8.4.2 Instalación de la tarjeta de control. La tarjeta de control Arduino requiere de tres conexiones con el banco de experimentación:

- *Conexión de energía.* Para la alimentación de energía se utiliza un adaptador AC-a-DC de 12 VDC y 1.5 amp, conector macho de 2.1mm con centro positivo y conector hembra en la placa (ver figura 114).

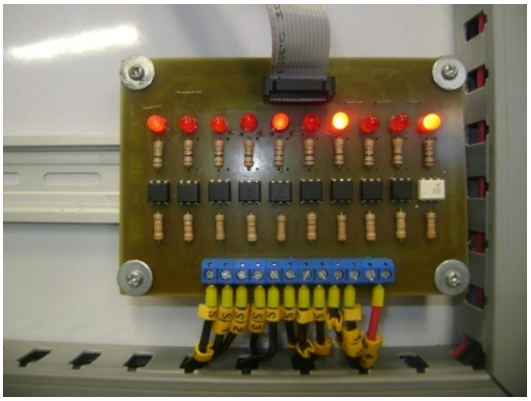
Figura 114. Conexiones de la tarjeta de control.



Fuente: Autores.

- *Conexión con tarjeta de interface de señales de entrada*. La conexión se realiza mediante un cable tipo Bus de datos (ver figura 115).

Figura 115. Conexión con tarjeta de interface de señales de entrada



Fuente: Autores.

- *Conexión con tarjeta de potencia*. La conexión se realiza mediante un cable tipo Bus de datos (Ver figura 116).

Figura 116. Conexión con tarjeta de potencia.

Fuente. Autores.

8.5 PRACTICA 1: ARRANQUE DEL MOTOR POR MEDIO DE EL SISTEMA DE ESTRELLA TRIANGULO.

8.5.1 Objetivos

- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo y el primer nivel de presión, utilizando el lenguaje Ladder aplicado al PLC.
- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo y el primer nivel de presión, utilizando el lenguaje C++ aplicado al Microcontrolador.
- Adquirir habilidades en la interpretación de planos eléctricos.

8.5.2 Fundamentación previa

- Revisar manual teórico.
- Identificar las principales características de los equipos utilizados en la elaboración de la práctica.

8.5.3 Procedimientos de seguridad

Antes de la elaboración de las prácticas se deben tener previo conocimiento de los procedimientos de seguridad que se deben seguir.

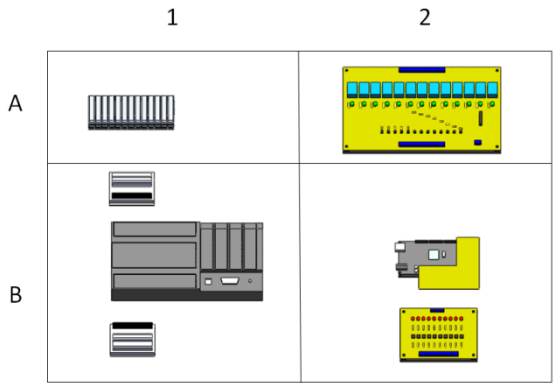
8.5.4 Ubicación e identificación de los componentes. Identifique cada uno de los componentes mencionados en la tabla 13, Para ver la distribución de los elementos ubicados en el panel (ver la figura 117).

Tabla 13. Componentes requeridos para la elaboración de la práctica uno.

ELEMENTO	NOMENCLATURA	UBICACIÓN
Parada de emergencia	-S20	Tablero de mando
Selector de energía principal	-S21	Tablero de mando
Indicador de energía principal	-H1	Tablero de mando
Selector de equipo (PLC o Microcontrolador)	-S22	Tablero de mando
Indicador del PLC	-H5	Tablero de mando
Indicador de Microncontrolador	-H6	Tablero de mando
Pulsador inicio del motor	-S10	Tablero de mando
Pulsador apagado del motor	-S11	Tablero de mando
Indicador motor encendido	-H2	Tablero de mando
PLC		B1
Conectores de entradas al PLC	C1	B1
Conectores de salidas del PLC	C2	B1
Microcontrolador		B2
Tarjeta de interface de entadas		B1
Tarjeta de potencia		A2
Bornes portafusibles	-F	A1
Microcontrolador		B2
Contactores arranque del motor	-KM1, -KM2, -KM3	
Relevos para arranque del motor	-K10, -K11 -KA10, -KA11	Tarjeta de potencia Tablero de potencia
Válvula de primer nivel de presión	V12	Unidad hidráulica

Fuente: Autores.

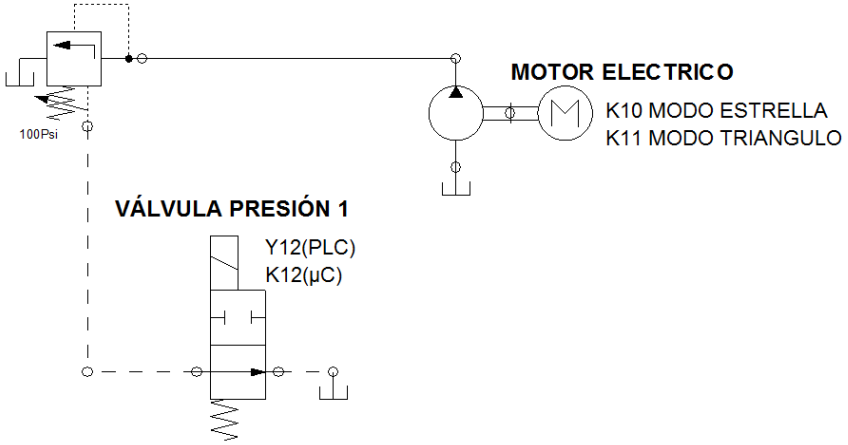
Figura 117. Distribución del los elementos en el panel .



Fuente. Autores.

8.5.5 Reconocimiento del circuito hidráulico. El circuito hidráulico a controlar consta de un motor eléctrico y una electroválvula de presión. La figura 118 ilustra el circuito hidráulico.

Figura 118. Circuito hidráulico de la primera práctica.

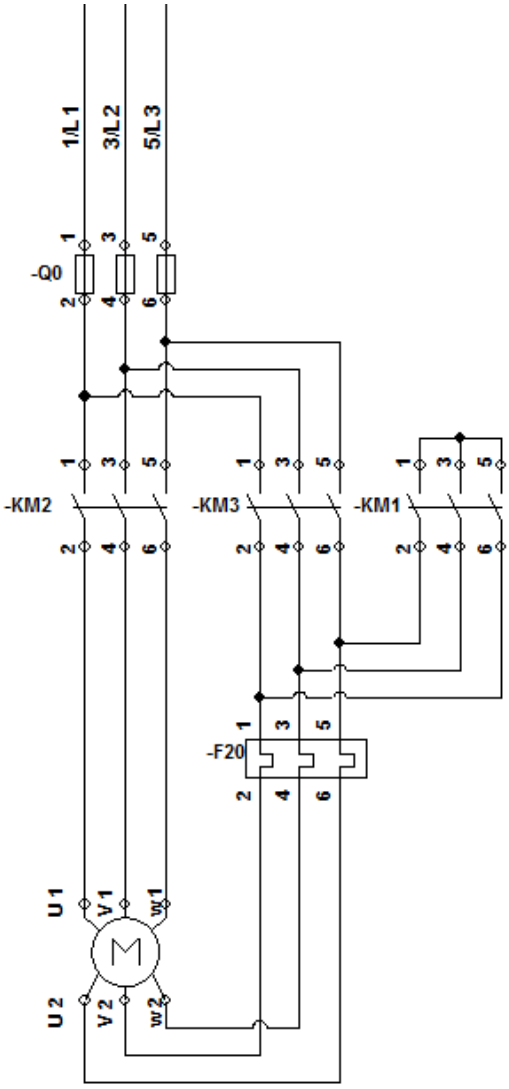


Fuente: Autores.

8.5.6 Reconocimiento de los circuitos eléctricos. Analice y discuta cada uno de los circuitos eléctricos requeridos para la elaboración de la práctica.

En la figura 119 se puede observar el diagrama eléctrico del arranque de un motor mediante el sistema de Estella-triangulo.

Figura 119. Esquema eléctrico del arrancador del motor.

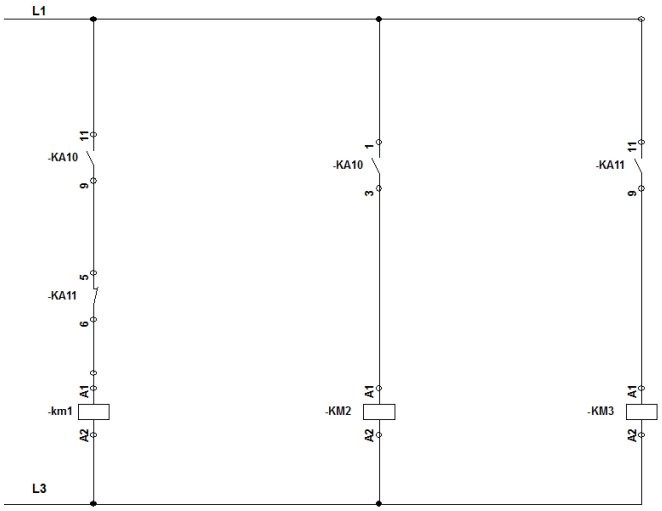


Fuente: Autores.

Inicialmente se acciona KM1 y KM2 para arrancar en el modo estrella, y posteriormente se desactiva KM1 y se activa KM3 para funcionar simultáneamente con KM2 y accionar el estado de triangulo, en esta transición KM2 nunca se desactiva.

La figura 120 muestra el esquema eléctrico del accionamiento de los contactores del motor requeridos para encenderlo.

Figura 120. Esquema eléctrico para el accionamiento de los contactores del motor.

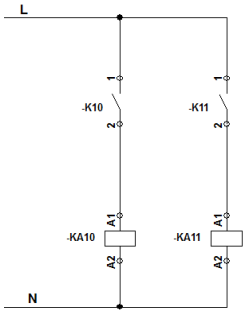


Fuente. Autores.

Inicialmente el relevo auxiliar KA10 acciona los contactores KM1 y KM2, posteriormente se acciona el relevo auxiliar KA11 para desactivar KM1 y activar KM3, que funciona simultáneamente con KM2.

Los relevos K10 y K11 que están ubicados en la tarjeta de potencia accionan sus respectivos relevos auxiliares KA10 y KA11 (ver figura 121).

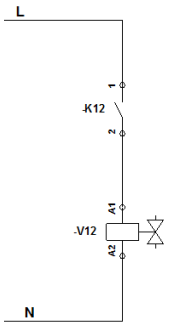
Figura 121. Esquema eléctrico para el accionamiento de los relevos auxiliares.



Fuente: Autores.

Al activar el relvo K12 se energiza el solenoide de la válvula del primer nivel de presión, tal como se muestra en la figura 122.

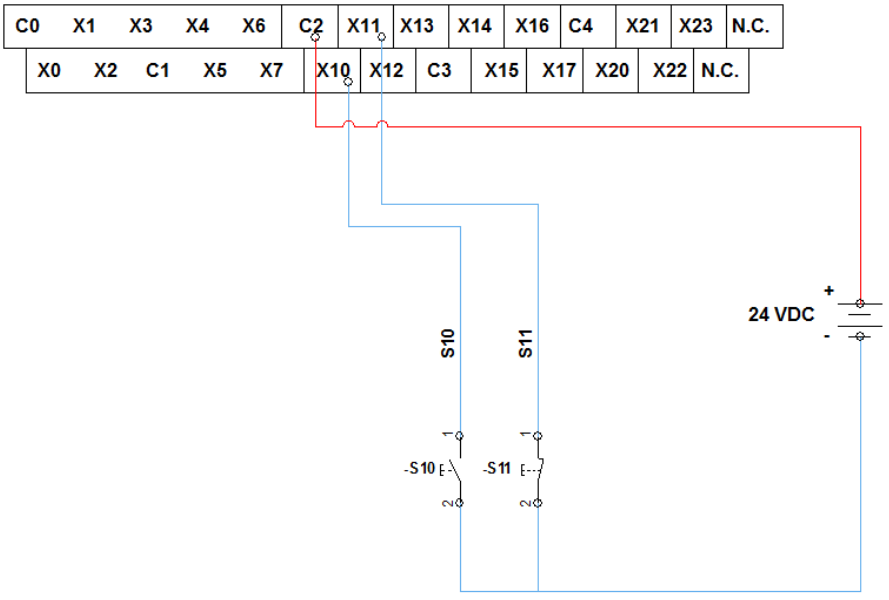
Figura 122. Esquema eléctrico de la válvula del primer nivel de presión.



Fuente: Autores.

Al presionar el pulsador S10 el cual es normalmente abierto se arranca el motor, y por medio de S11 el cual es un contacto normalmente cerrado se detiene el motor. Las conexiones de entradas del PLC se pueden observar en la figura 123.

Figura 123. Esquema eléctrico de las conexiones de entrada del PLC requeridas para la práctica.

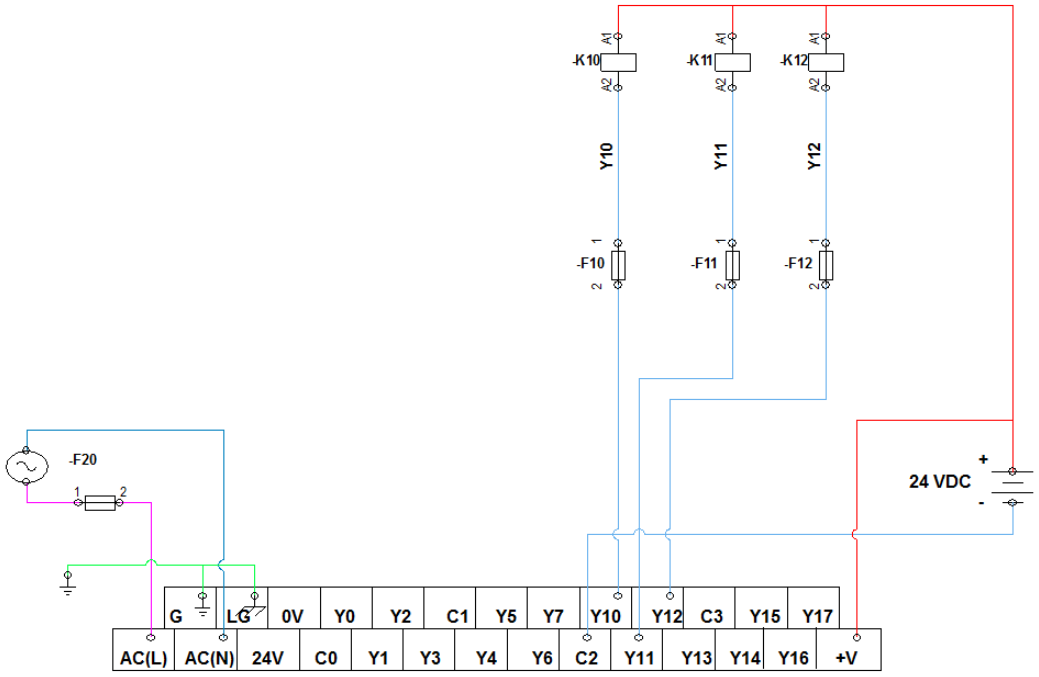


Fuente: Autores.

La salida Y10 acciona el relevo k10 el cual arranca el motor en modo estrella posteriormente la salida Y11 activa el relevo k11 para pasar el motor al modo triangulo.

La salida Y12 acciona el relevo K12 el cual energiza el solenoide V12 que corresponde al primer nivel de presión. Las conexiones de salidas se pueden observar en la figura 124.

Figura 124. Esquema eléctrico de las conexiones de salida del PLC requeridas para la práctica.



Fuente: Autores.

La tabla 14 muestra los puertos de salidas del Microncontrolador y los relevos que activan en la tarjeta de potencia, para la elaboración de la práctica.

Tabla 14. Puertos de salidas de Microcontrolador y los relevos que activan en la tarjeta de potencia.

PUERTOS DIGITALES DE LA TARJETA ARDUINO	SEÑALES DE SALIDA-TARJETA DE POTENCIA
31	K12
33	K11
35	K10

Fuente: autores.

La tabla 15 muestra los puertos de entrada del Microncontrolador y las señales de los sensores que llegan a estos puertos.

Tabla 15. Puertos de entrada del Microncontrolador y señales de los sensores que llegan a los mismos.

PUERTOS ANALÓGICOS DE LATARJETA ARDUINO	SEÑALES DE ENTRADA-TARJETA INTERFAZ
A6	S10
A7	S11

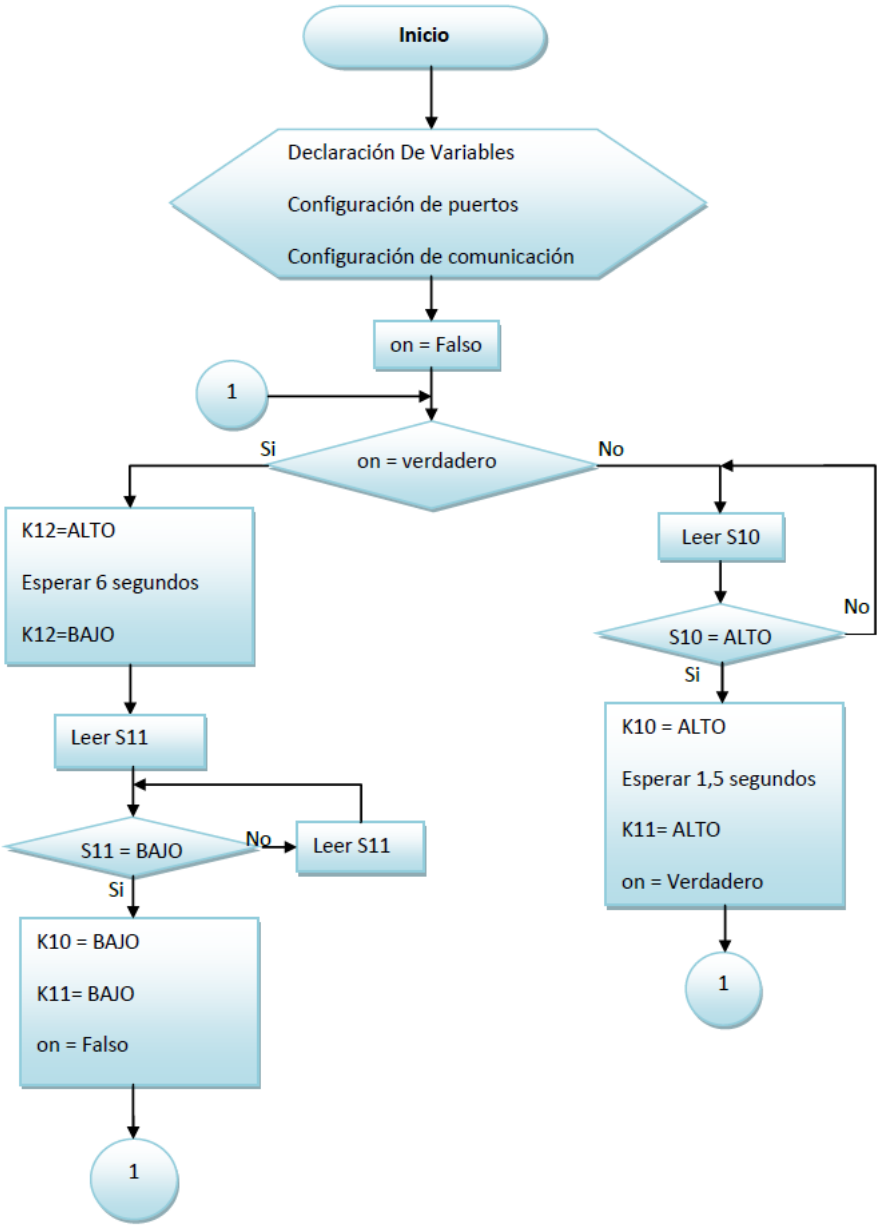
Fuente: Autores.

8.5.7 Programación. Programe la siguiente secuencia para el PLC y el Microncontrolador:

a) Al presionar S10 se enciende motor por medio del relevo K10.
b) 1.5 seg después encender el relevo K11 para pasar de estrella a triangulo.
c) 1 seg después active el primer nivel de presión por medio del relevo K12 y manténgalo 6 segundos.
d) Al presionar S11 apagar motor.

En la figura 125 se muestra un ejemplo del diagrama de flujo del programa para la primera práctica.

Figura 125. Diagrama de flujo del programa de la primera práctica.



Fuente: Autores.

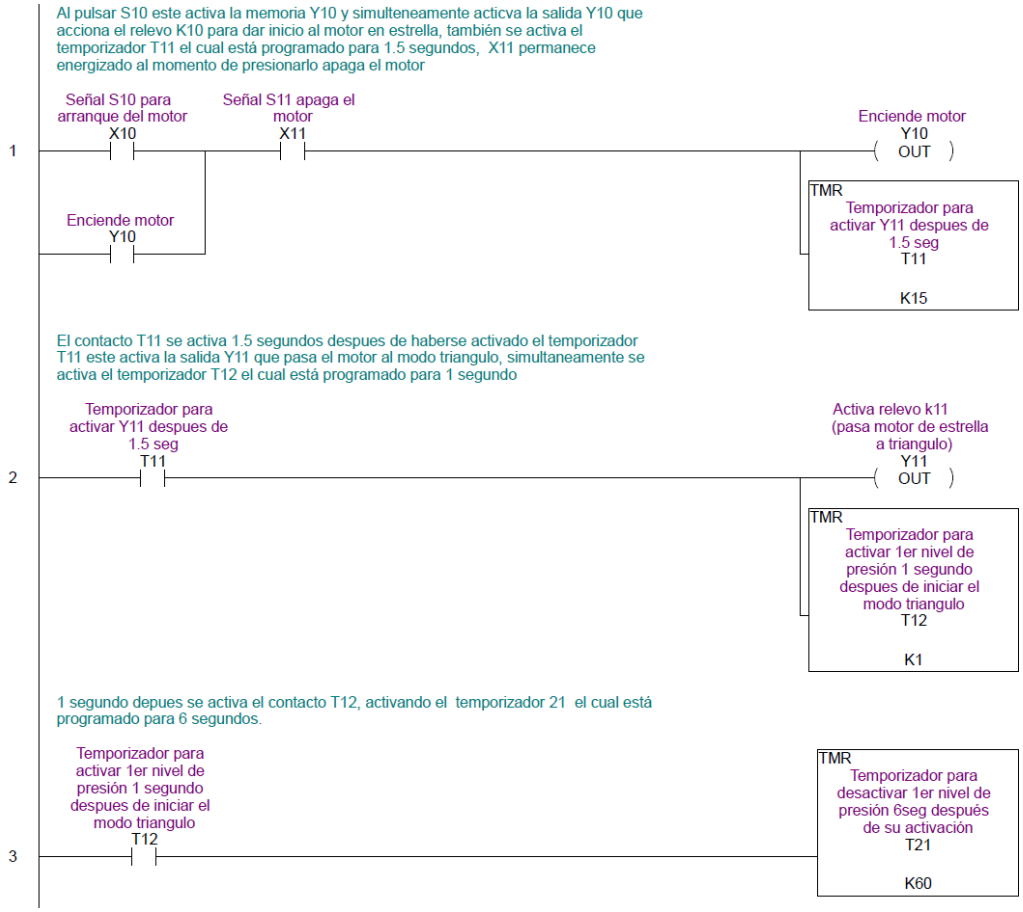
A. PLC

- Antes de realizar el programa lea la guía de programación para el PLC.
- Utilice el software DirectSOFT 5 para elaborar el programa.

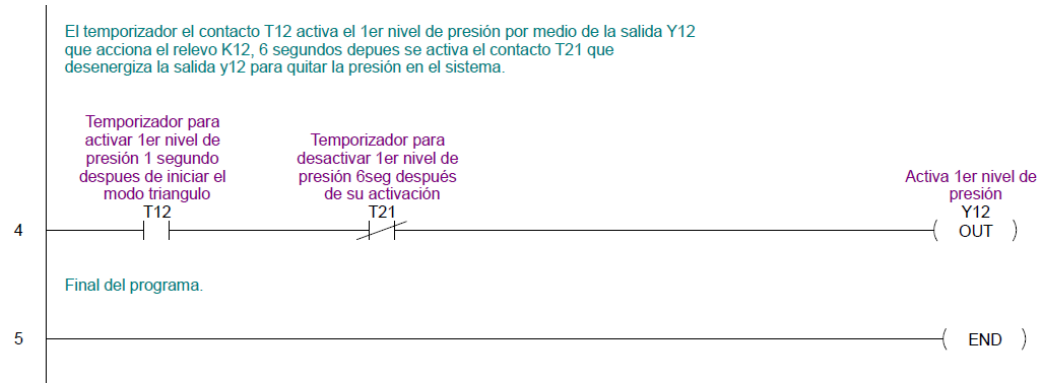
Ejemplo de programación.

En la figura 125 se muestra un ejemplo del programa para el PLC de la primera práctica.

Figura 126. Ejemplo de programación del PLC de la primera práctica.



Fuente: Autores.



B. Microcontrolador

- Antes de realizar el programa lea la guía de programación para el Microcontrolador.
- Utilice el software Arduino para elaborar el programa

Ejemplo de programación.

```
/*  Ejemplo  primera practica
Prender motor, probar la presión y venteo apagar moto
 NOTA:  *digitar el número de ciclos(cic) o cero(0) para repetir ciclos
indefinidamente
 */
//definición de variables
int var1;             //variables de lectura  de la señal S10 (encendido de
motor)
int var2;
int var3;            //variables de lectura  de la señal S11 (apagar motor)
int var4;

int S10 = A6;      //   pulsador encendido
int S11 = A7;      // pulsador apagado
int senal_ens;     //  almacena  señal encendido
int senal_apag;  // almacena  señal apagado
int on;                  //  estado del motor
int k10 = 35;      // relevo del encendido modo estrella
int k11 = 33;      // relevo del apagado modo triangulo
int k12 = 31;        // 1er nivel de presión
int cont1;   // contador de ciclos
int cic;      // número de siclos
int cont3;   //contador para impresión de número de ciclos

//definición de variables

void setup(){
  Serial.begin(9600);         // comunicación
  cic = 2;   // en esta variable se digita el número de ciclos
  pinMode(S10, INPUT);    //define como entrada
```

```
  pinMode(k10, OUTPUT);   //define como salida
  pinMode(k11, OUTPUT);   //define como salida
  var1 = 0;
  var2 = 0;
  on = 0;
  senal_ens = 0;          // normalmente abierto
  pinMode(S11, INPUT);    //define como entrada
  var3 = 0;
  var4 = 0;
  senal_apag = 1;          // normalmente cerrado
  pinMode(k12, OUTPUT);   //define como salida
}
void loop(){     //código de programa
  if(on == 0){           // si el motor está apagado entra
    var1 = analogRead(S10);  // lee estado de señal y almacena en var1
    if (var1 >= 500){      // convierte señal en cero o uno
      var1=1;
    }
    else {
      var1=0;
    }
    delay(10);             //retraso de 10 milisegundos
    var2 = analogRead(S10);  // lee estado de señal y almacena en var2
    if (var2 >= 500){      // convierte señal en cero o uno
      var2=1;
    }
    else {
      var2=0;
    }
    if (var1 == var2){      // evita rebote de contacto
      if (var1 != senal_ens){        // selecciona cuando hay  cambio de la señal
        if (var1 == LOW){    // selecciona cuando se suelta el pulsador
          digitalWrite(k10, HIGH);      //prende el motor modo estrella
          Serial.println("motor prendido modo estrella");   //imprime "motor
prendido modo estrella"
          delay(1500);       // espera 1,5 segundos
          digitalWrite(k11, HIGH);     //   pasa a modo triangulo
          Serial.println("motor prendido modo triangulo");   //imprime "motor
prendido modo triangulo"
          on = 1;
          delay(1000);     // retraso de un segundo
          cont1 = 0;
          cont3 = 0;
        }
        senal_ens = var1;
      }
    }
  }

  while(cont1 < cic && on ==1){   // siempre que el contador de ciclos sea menor
que la cantidad de ciclos requeridos hace...
    cont3++;     // suma una unidad a el contador de ciclos para la impresión
    Serial.print("ciclo número ");
    Serial.println(cont3);        //   imprime en número de ciclo
    digitalWrite(k12, HIGH);     //  primer nivel de presión
    Serial.println("primer nivel de presión");  //  imprime primer nivel de
presión
    delay(6000); // retraso de 6 segundos
    digitalWrite(k12, LOW);     //  primer nivel de presión
    Serial.println("venteo");  //  imprime venteo
    delay(2000);
```

```
    if(cic != 0){ // si el número de ciclos es cero el ciclo se repetirá hasta
que se apague el motor
      cont1++;   //suma una unidad al número de ciclos
    }
  }

  if(on == 1){           // si el motor está encendido entra
    var3 = analogRead(S11);  // lee estado de señal y almacena en var3
    if (var3 >= 500){      // convierte señal en cero o uno
      var3=1;
    }
    else {
      var3=0;
    }
    delay(10);             //retraso de 10 milisegundos
    var4 = analogRead(S11);  // lee estado de señal y almacena en var4
    if (var4 >= 500){      // convierte señal en cero o uno
      var4=1;
    }
    else {
      var4=0;
    }
    if (var3 == var4){      // evita rebote de contacto
      if (var3 != senal_apag){        // selecciona cuando hay  cambio de la
señal
        if (var3 == HIGH){    // selecciona cuando se suelta el pulsador
          digitalWrite(k10, LOW);    // apaga motor
          digitalWrite(k11, LOW);
          Serial.println("motor apagado");   //imprime "motor apagado"
          on = 1;
        }
        senal_apag = var3;
      }
    }
  }
}
```

8.5.8 Evaluación para los estudiantes.

1. ¿En qué consiste el sistema de arranque Estrella-Triangulo de un motor trifásico?
2. Elabore un programa utilizando el lenguaje Ladder para el PLC y en C++ para el Microcontrolador que permita arrancar un motor trifásico con el sistema Estrella-Triangulo.
3. ¿Cuál es el voltaje de las señales de entrada al PLC?
4. ¿Donde están ubicados los relevos K10 y K11?

8.6 PRÁCTICA 2: ARRANQUE DEL MOTOR Y ACCIONAMIENTO DEL ACTUADOR 1.

8.6.1 Objetivos

- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo utilizando el lenguaje Ladder aplicado al PLC.
- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo utilizando el lenguaje C++ aplicado al Microcontrolador.
- Programar el accionamiento del actuador 1 utilizando el lenguaje Ladder aplicado al PLC.
- Programar el accionamiento del actuador 1 utilizando el lenguaje C++ aplicado al Microcontrolador.

8.6.2 Fundamentación previa

- Revisar manual teórico.
- Identificar las principales características de los equipos utilizados en la elaboración de la práctica.

8.6.3 Procedimientos de seguridad. Antes de la elaboración de las prácticas se deben tener previo conocimiento de los procedimientos de seguridad que se deben seguir.

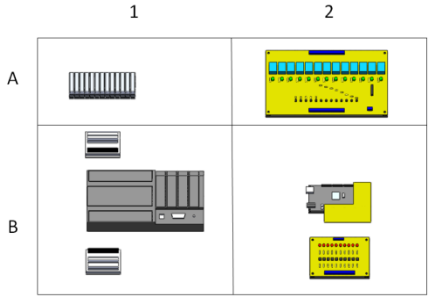
8.6.4 Ubicación e identificación de los componentes. Identifique cada uno de los componentes mencionados en la tabla 16. Para ver la distribución de los elementos ubicados en el panel ver la figura 127.

Tabla 16. Componentes requeridos para la elaboración de la práctica dos.

ELEMENTO	NOMENCLATURA	UBICACIÓN
Parada de emergencia	-S20	Tablero de mando
Selector de energía principal	-S21	Tablero de mando
Indicador de energía principal	-H1	Tablero de mando
Selector de equipo (PLC o Microcontrolador)	-S22	Tablero de mando
Indicador del PLC	-H5	Tablero de mando
Indicador de Microncontrolador	-H6	Tablero de mando
Pulsador inicio del motor	-S10	Tablero de mando
Pulsador apagado del motor	-S11	Tablero de mando
Indicador motor encendido	-H2	Tablero de mando
Indicador de válvula de presión 1	-H3	Tablero de mando
PLC		B1
Conectores de entradas al PLC	C1	B1
Conectores de salidas del PLC	C2	B1
Microcontrolador		B2
Tarjeta de interface de entadas		B1
Tarjeta de potencia		A2
Bornes portafusibles	-F	A1
Microcontrolador		B2
Contactores arranque del motor	-KM1, -KM2, -KM3	
Relevos para arranque del motor	-K10, -K11 -KA10, -KA11	Tarjeta de potencia Tablero de potencia
Relevo para activación de 1er nivel de presión	-K12	Tarjeta de potencia
Relevo para activación solenoide V1-0 de la válvula 1	-k0	Tarjeta de potencia
Relevo para activación solenoide V1-1 de la válvula 1	-k1	Tarjeta de potencia
Electroválvula 2		Unidad hidráulica
Sensor 0	-S0	Inicio de carrera del actuador 1
Sensor 1	-S1	Final de carrera del actuador 1
Válvula primer nivel de presión	-S12	Unidad hidráulica
Válvula 1	V1	Unidad hidráulica
Solenoides válvula 1	V1-0 (sale actuador) V1-1 (entra actuador)	Unidad hidráulica
Actuador 1		Unidad hidráulica
Válvula de 1er nivel de presión	V12	Unidad hidráulica
Alarma	H9	Tablero eléctrico

Fuente: Autores.

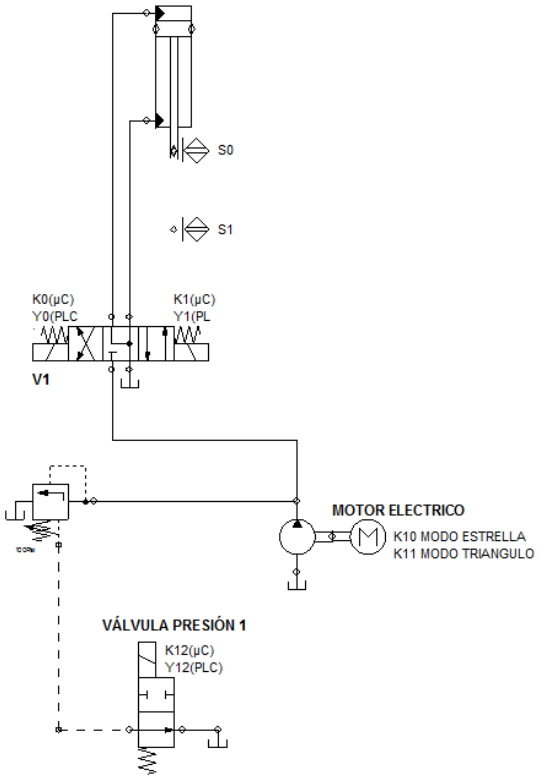
Figura 127. Distribución del los elementos en el panel.



Fuente Autores.

8.6.5 Reconocimiento del circuito hidráulico. El circuito hidráulico a controlar consta de un motor eléctrico, una electroválvula de presión y un actuador controlado por una electroválvula. La figura 128 ilustra el circuito hidráulico.

Figura 128. Circuito hidráulico de la segunda práctica.



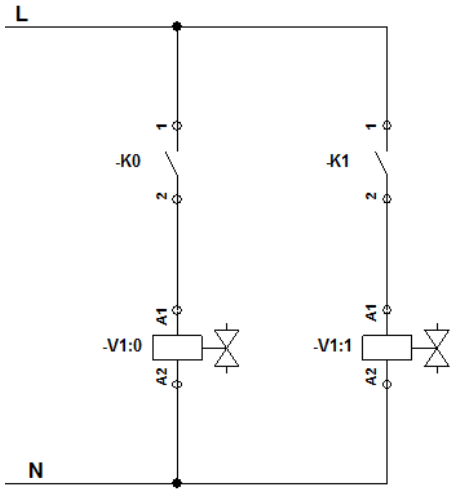
Fuente: Autores.

8.6.6 Reconocimiento de los circuitos eléctricos. Analice y discuta cada uno de los circuitos eléctricos requeridos para la elaboración de la práctica 2.

Ver figura 119, 120, 121, 122.

El relevo K0 acciona el solenoide V1-0 que corresponde a la válvula del actuador 1 y lo mueve hacia abajo. El relevo K1 acciona el otro solenoide V1-1 para regresarlo a la posición inicial. Esto se puede observar el la figura 129.

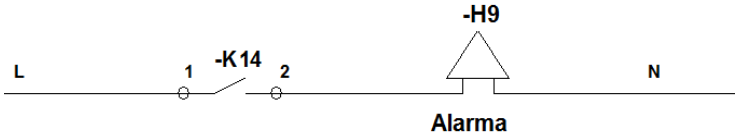
Figura 129. Esquema eléctrico de accionamiento de la electroválvula 1.



Fuente. Autores.

La figura 130 ilustra el esquema de accionamiento de la alarma, la cual debe sonar en caso de no haber señal por parte de alguno de los sensores.

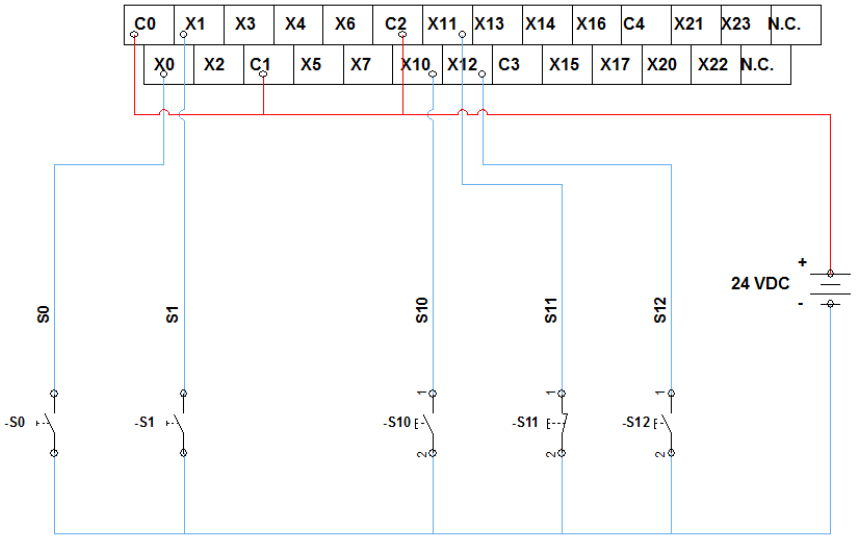
Figura 130. Esquema de accionamiento de la alarma.



Fuente Autores.

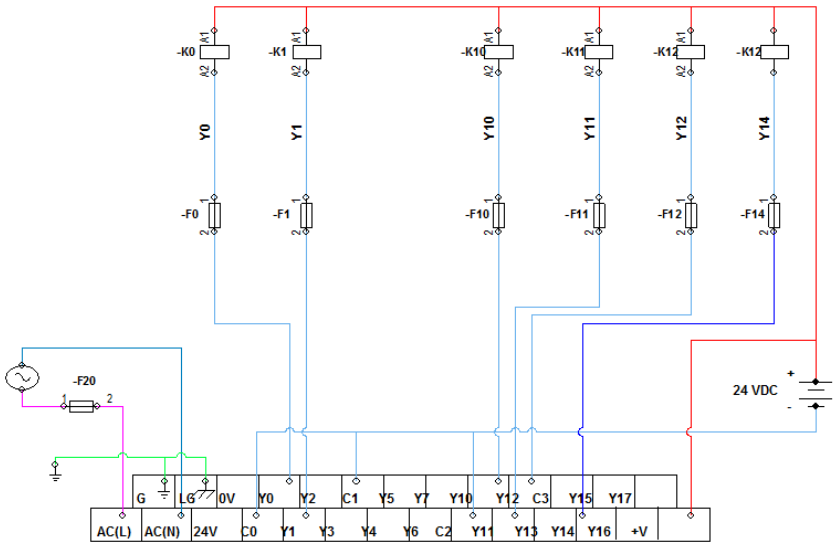
En la figura 131 y 132 se observan las conexiones de entradas y salidas respectivamente del PLC que se requieren para la elaboración de la práctica 2.

Figura 131. Esquema eléctrico de las conexiones de entrada del PLC requeridas para la segunda práctica.



Fuente. Autores.

Figura 132. Esquema eléctrico de las conexiones de salida del PLC requeridas para la segunda práctica.



Fuente. Autores.

La tabla 17 muestra los puertos de salidas del Microncontrolador y los relevos que activan en la tarjeta de potencia, para la elaboración de la práctica.

Tabla 17. Puertos de salidas de Microcontrolador y los relevos que activan en la tarjeta de potencia (Práctica dos).

PUERTOS DIGITALES DE LA TARJETA ARDUINO	SEÑALES DE SALIDA-TARJETA DE POTENCIA
27	K14
31	K12
33	K11
35	K10
49	K1
51	K0

Fuente: autores.

La tabla 18 muestra los puertos de entrada del Microncontrolador y las señales de los sensores que llegan a estos puertos.

Tabla 18. Puertos de entrada del Microncontrolador y señales de los sensores que llegan a los mismos (Práctica dos).

PUERTOS ANALÓGICOS DE LATARJETA ARDUINO	SEÑALES DE ENTRADA-TARJETA INTERFAZ
A6	S10
A7	S11
A8	S0
A9	S1

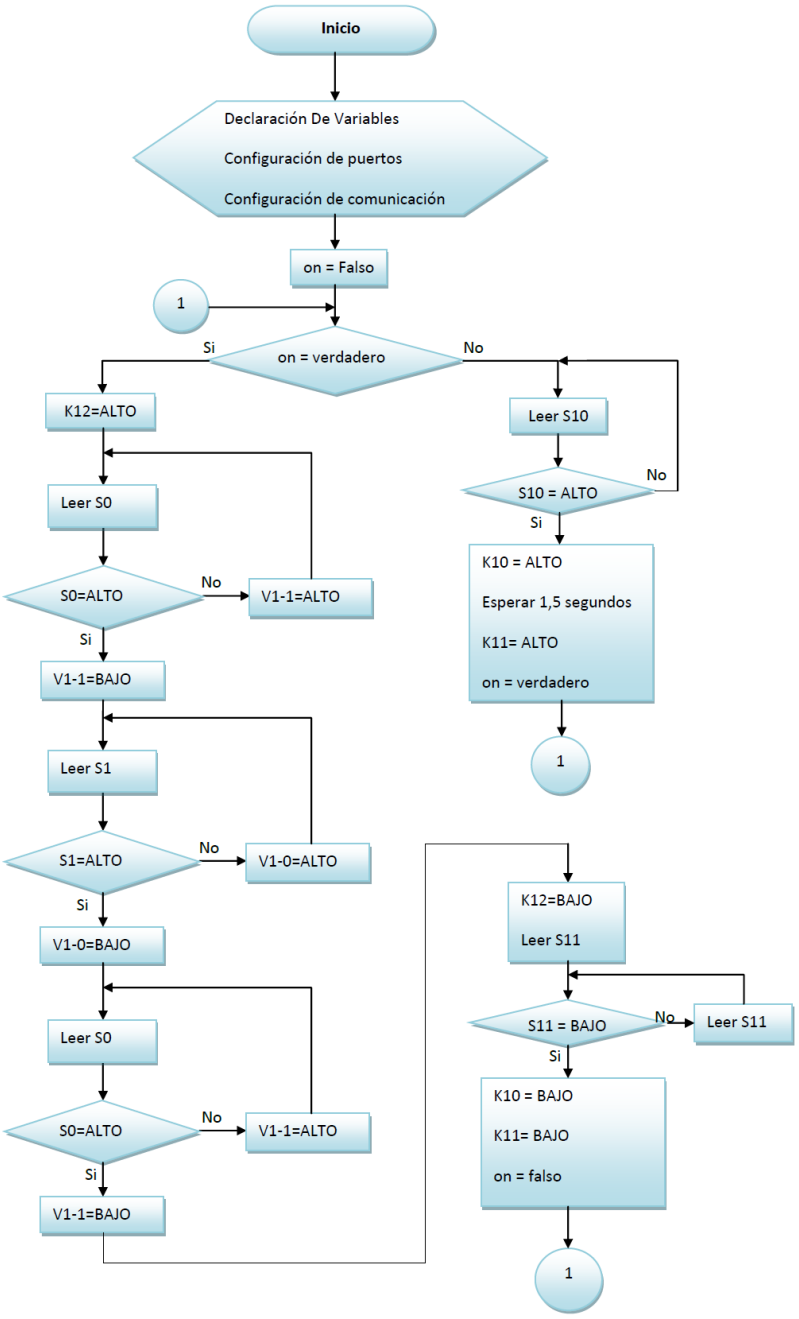
Fuente: Autores.

8.6.7 Programación. Programe la siguiente secuencia para el PLC y el Microncontrolador.

a) Al presionar S10 se enciende motor por medio del relevo K10.
b) 1.5 seg después encender el relevo K11 para pasar de estrella a triangulo.
c) 1 seg después active el primer nivel de presión por medio del relevo K12.
d) Compruebe que el actuador 1 se encuentra en posición correcta.
e) Realice la siguiente secuencia:
    1. Active solenoide V1-0 para sacar el actuador 1.
    2. Sostenerlo abajo por 5 segundos
    3. Active solenoide V1-1 para regresar a su posición inicial el actuador 1
    4. Desactive el primer nivel de presión.
    5. Apague motor.
f) Programe la alarma de tal forma que se active si los sensores no envían señales. tenga en cuenta que el tiempo que se demora el actuador en ir de un extremo a otro es de 6 segundos.

En la figura 133 se muestra un ejemplo del diagrama de flujo del programa para la segunda práctica.

Figura 133. Diagrama de flujo del programa de la segunda práctica.



Fuente: Autores.

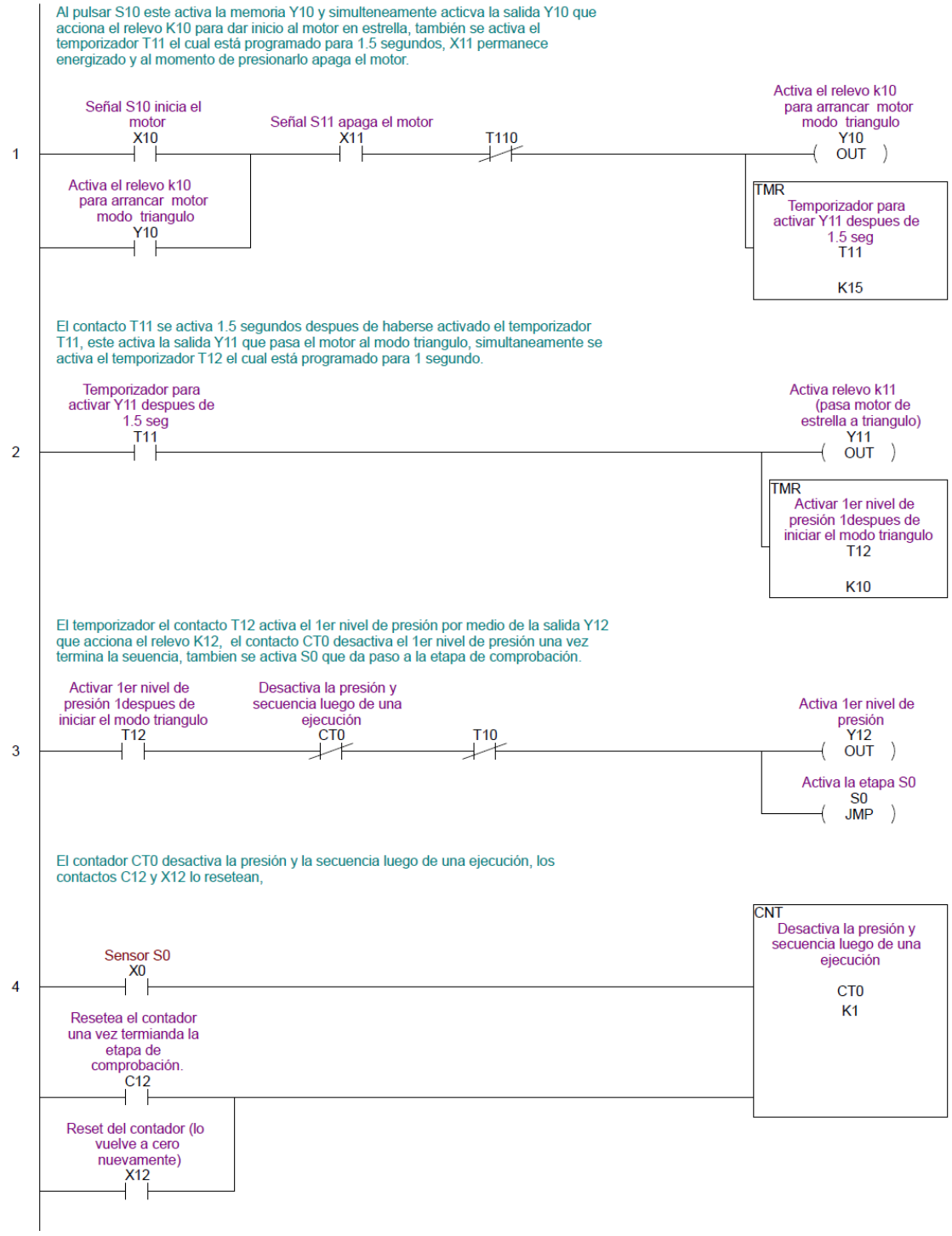
A. PLC

- Antes de realizar el programa lea la guía de programación para el PLC.
- Utilice el software DirectSOFT 5 para elaborar el programa.

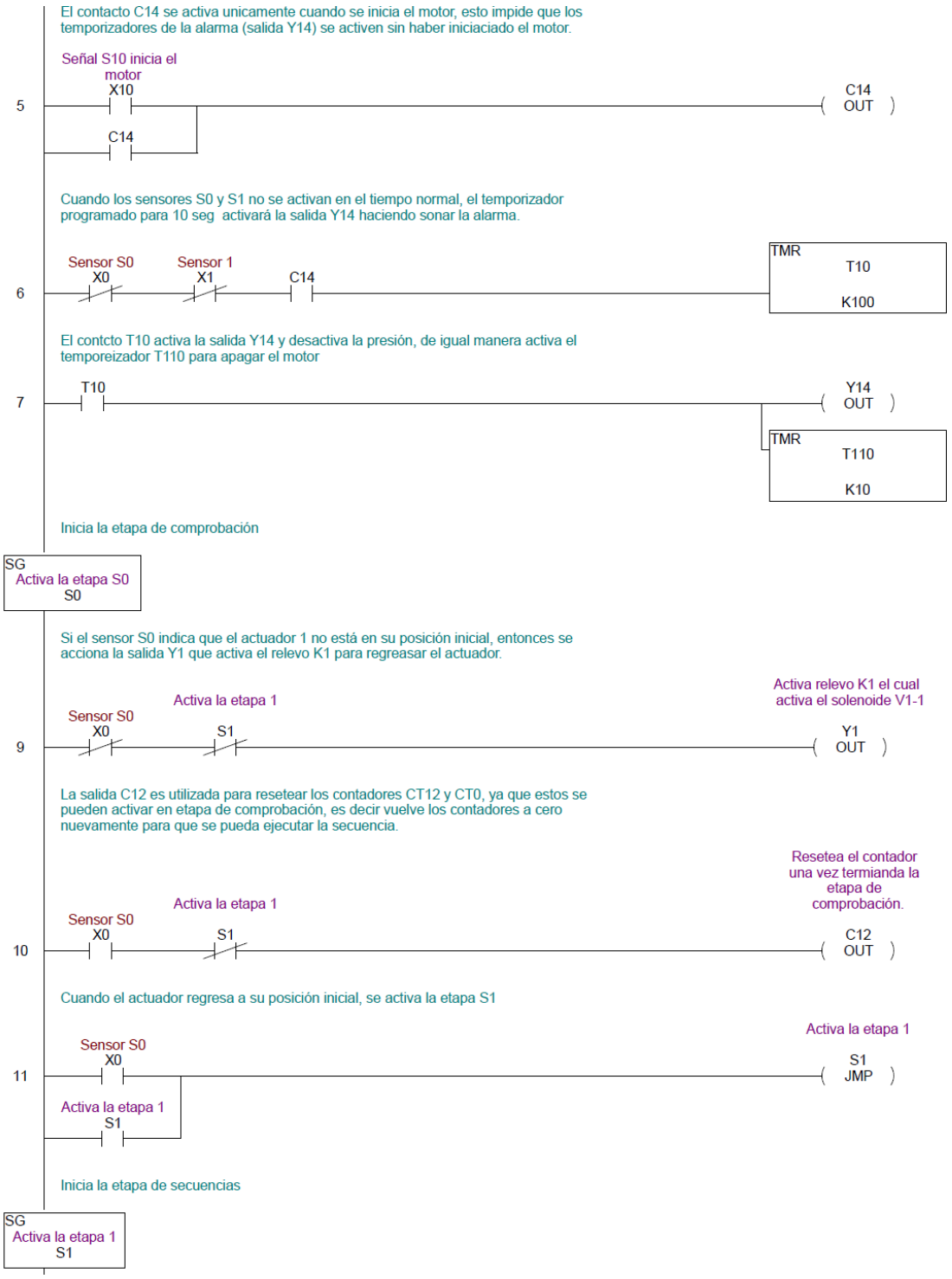
Ejemplo de programación.

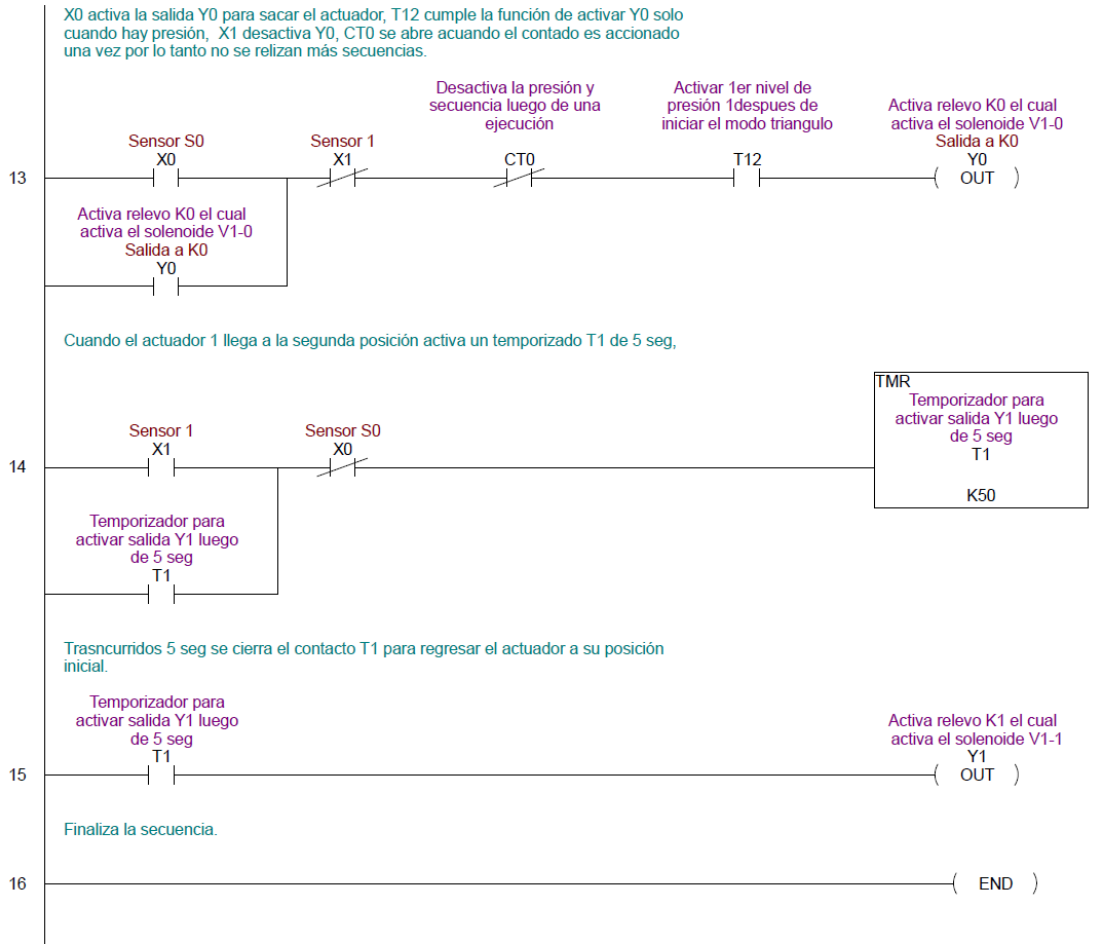
En la figura 134 se muestra un ejemplo de programación para el PLC de la segunda práctica.

Figura 134. Ejemplo de programación del PLC de la segunda práctica.



Fuente: Autores.


El contacto C14 se activa unicamente cuando se inicia el motor, esto impide que los temporizadores de la alarma (salida Y14) se activen sin haber iniciaciado el motor.
Señal S10 inicia el motor
X10
C14
5
C14 OUT
Cuando los sensores S0 y S1 no se activan en el tiempo normal, el temporizador programado para 10 seg activará la salida Y14 haciendo sonar la alarma.
Sensor S0
X0
Sensor 1
X1
C14
6
TMR
T10
K100
El contcto T10 activa la salida Y14 y desactiva la presión, de igual manera activa el temporeizador T110 para apagar el motor
T10
7
Y14 OUT
TMR
T110
K10
Inicia la etapa de comprobación
SG
Activa la etapa S0
S0
Si el sensor S0 indica que el actuador 1 no está en su posición inicial, entonces se acciona la salida Y1 que activa el relevo K1 para regreasar el actuador.
Activa la etapa 1
Activa relevo K1 el cual activa el solenoide V1-1
Sensor S0
X0
S1
9
Y1 OUT
La salida C12 es utilizada para resetear los contadores CT12 y CT0, ya que estos se pueden activar en etapa de comprobación, es decir vuelve los contadores a cero nuevamente para que se pueda ejecutar la secuencia.
Resetea el contador una vez termianda la etapa de comprobación.
Activa la etapa 1
Sensor S0
X0
S1
10
C12 OUT
Cuando el actuador regresa a su posición inicial, se activa la etapa S1
Activa la etapa 1
Sensor S0
X0
11
S1 JMP
Activa la etapa 1
S1
Inicia la etapa de secuencias
SG
Activa la etapa 1
S1


X0 activa la salida Y0 para sacar el actuador, T12 cumple la función de activar Y0 solo cuando hay presión, X1 desactiva Y0, CT0 se abre acuando el contado es accionado una vez por lo tanto no se relizan más secuencias.
Desactiva la presión y secuencia luego de una ejecución
Activar 1er nivel de presión 1despues de iniciar el modo triangulo
Activa relevo K0 el cual activa el solenoide V1-0 Salida a K0
Sensor S0
X0
Sensor 1
X1
CT0
T12
Y0
OUT
13
Activa relevo K0 el cual activa el solenoide V1-0 Salida a K0
Y0
Cuando el actuador 1 llega a la segunda posición activa un temporizado T1 de 5 seg,
TMR
Temporizador para activar salida Y1 luego de 5 seg
T1
K50
Sensor 1
X1
Sensor S0
X0
14
Temporizador para activar salida Y1 luego de 5 seg
T1
Trasncurridos 5 seg se cierra el contacto T1 para regresar el actuador a su posición inicial.
Temporizador para activar salida Y1 luego de 5 seg
T1
Activa relevo K1 el cual activa el solenoide V1-1
Y1
OUT
15
Finaliza la secuencia.
END
16

B. Microcontrolador

- Antes de realizar el programa lea la guía de programación para el Microcontrolador.
- Utilice el software Arduino para elaborar el programa.

Ejemplo de programación.

```
/*  Ejemplo  segunda practica
 Cilindro número 1 ( +, retraso 5seg, - )
 NOTA:  *digitar el número de ciclos(cic) o cero(0) para repetir ciclos
indefinidamente
 *digitar los tiempos de retraso (t1mas, t1men)
 */
int cic = 3;
int t1mas = 3;     //tiempo de permanencia cilindro 1 +(segundos)
int t1men = 0;        //tiempo de permanencia cilindro 1 -(segundos)
//variables
int var1=0;         //variables de lectura  de la señal S10 (encendido de motor)
int var2=0;
int var3=0;      //variables de lectura  de la señal S11 (apagar motor)
int var4=0;

int S10 = A6;     // pulsador encendido entrada analógica  6
int S11 = A7;     // pulsador apagado entrada analógica  7
int senal_ens=0;    // almacena  señal encendido
int senal_apag=1;  // almacena  señal apagado
int on = 0;           // estado del motor
int K10 = 35;     // relevo del encendido modo estrella pin 35
int K11 = 33;     // relevo del apagado modo triangulo pin 33
int K12 = 31;       // 1er nivel de presión pin 31
int K14 = 27;      // alarma
int cont1=0;   // contador de ciclos
int cont2=0;   // contador para actuadores en posición inicial
int cont3=0;   //contador para impresión de número de ciclos
int Tref;    // tiempo de referencia para la alarma

// cilindro 1
int S0 = A8;     //sensor 0 cilindro 1 ... entrada analógica pin 8
int S1 = A9;     //sensor cilindro 1 ... entrada analógica pin 9
int K0 = 51;     //válvula cilindro 1 +
int K1 = 49;     //válvula cilindro 1 -
int lS0 = 0;
int lS1 = 0;

void setup(){
  Serial.begin(9600);       // comunicación

  pinMode(K10, OUTPUT);   //define como salida estrella
  pinMode(K11, OUTPUT);   //define como salida triangulo

  pinMode(K12, OUTPUT);   //define como salida presión 1

  pinMode(K14, OUTPUT);   //define como salida la alarma
```

```
  //cilindro 1
  pinMode(K0, OUTPUT);   //define como salida electroválvula cil 1 +
  pinMode(K1, OUTPUT);   //define como salida electroválvula cil 1 -

}
void loop(){
  encender();
  practica2();
  apagar();
}
void encender(){
  if(on == 0){           // si el motor está apagado entra
    var1 = analogRead(S10);  // lee estado de señal y almacena en var1
    if (var1 >= 500){      // convierte señal en cero o uno
      var1=1;
    }
    else {
      var1=0;
    }
    delay(10);             //retraso de 10 milisegundos
    var2 = analogRead(S10);  // lee estado de señal y almacena en var2
    if (var2 >= 500){      // convierte señal en cero o uno
      var2=1;
    }
    else {
      var2=0;
    }
    if (var1 == var2){      // evita rebote de contacto
      if (var1 != senal_ens){        // selecciona cuando hay  cambio de la señal
        if (var1 == LOW){    // selecciona cuando se suelta el pulsador
          digitalWrite(K14, LOW);
          digitalWrite(K10, HIGH);      //prende el motor modo estrella
          Serial.println("motor prendido modo estrella");   //imprime "motor
prendido modo estrella"
          delay(1500);       // espera 1,5 segundos
          digitalWrite(K11, HIGH);     //   pasa a modo triangulo
          Serial.println("motor prendido modo triangulo");   //imprime "motor
prendido modo triangulo"
          on = 1;
          delay(1);     // retraso de un segundo
          cont1 = 0;
          cont2 = 0;
          cont3 = 0;
        }
        senal_ens = var1;
      }
    }
  }
}
void practica2(){
  if (cont2 < 1 && on == 1){
    digitalWrite(K12, HIGH);
    delay(700);
    Serial.println("nivel de presión 1");
    posicioni1();
    ciclos();
    digitalWrite(K12, LOW);
    if (on == 1){
      Serial.println("fin de la tarea");
      digitalWrite(K12, LOW);
      Serial.println("venteo");
```

```
    }
    cont2++;
  }
}
void posicioni1(){
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  if(on == 1){
  Serial.println("posición correcta cilindro 1");
  }
}
void ciclos(){
  while(cont1 < cic && on == 1){   // siempre que el contador de ciclos sea menor
que la cantidad de ciclos requeridos hace...
    cont3++;     // suma una unidad a el contador de ciclos para la impresión
    Serial.print("ciclo número ");
    Serial.println(cont3);        //   imprime en número de ciclo
    act_1_mas();
    act_1_men();
    if(cic != 0){ // si el número de ciclos es cero el ciclo se repetirá hasta
que se apague el motor
      cont1++;   //suma una unidad al número de ciclos
    }
  }

}
void act_1_mas(){
  lS1 = analogRead(S1);
  Tref = millis();
  while (lS1 <= 500 && on == 1){
    digitalWrite(K0, HIGH);
    lS1 = analogRead(S1);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K0, LOW);
  fdelay(t1mas);
}
void act_1_men(){
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  fdelay(t1men);
}

void apagar(){
  var3 = analogRead(S11);  // lee estado de señal y almacena en var3
  if (var3 >= 500){      // convierte señal en cero o uno
```

```
    var3=1;
  }
  else {
    var3=0;
  }
  delay(10);              //retraso de 10 milisegundos
  var4 = analogRead(S11);  // lee estado de señal y almacena en var4
  if (var4 >= 500){       // convierte señal en cero o uno
    var4=1;
  }
  else {
    var4=0;
  }
  if (var3 == var4){       // evita rebote de contacto
    if (var3 != senal_apag){        // selecciona cuando hay  cambio de la señal
      if (var3 == HIGH){
        digitalWrite(K14, LOW);
        if (on == 1){    // selecciona cuando se suelta el pulsador
          digitalWrite(K12, LOW);    // ventear
          digitalWrite(K13, LOW);    // ventear
          digitalWrite(K10, LOW);    // apaga motor
          digitalWrite(K11, LOW);
          Serial.println("motor apagado");   //imprime "motor apagado"
          on = 0;
        }
      }
      senal_apag = var3;
    }
  }
}
void fdelay(int s){
  int a = 0;
  while (a <= s*1000 && on == 1){
    a = a + 100;
    delay (100);
    apagar();
  }
}

void alarma(int Talarma, int act){
  int T = -(Tref - millis());
  if (abs(T) >= Talarma && on == 1){
    digitalWrite(K14, HIGH);
    digitalWrite(K12, LOW);
    digitalWrite(K13, LOW);
    delay(1000);
    digitalWrite(K10, LOW);
    digitalWrite(K11, LOW);
    on = 0;
    Serial.print("ALERTA: hay problemas en el actuador ");
    Serial.println(act);
    Serial.println("Se recomienda revisar las electroválvulas y los sensores de
dicho actuador");
  }
}
```

8.6.8 Evaluación para los estudiantes.

1. Elabore un programa utilizando el lenguaje Ladder para el PLC y en C++ para el Microcontrolador que permita realizar la secuencia indicada en el ítem de programación.
2. ¿Dónde se encuentran ubicados los sensores S0 y s1?
3. ¿Cuál es valor numérico del primer nivel de presión?
4. ¿Cuál es la tensión de alimentación de la tarjeta de control Arduino?

8.7 PRÁCTICA 3: ARRANQUE DEL MOTOR Y ACCIONAMIENTO DE LOS ACTUADORES 1 Y 2

8.7.1 Objetivos

- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo y el primer nivel de presión, utilizando el lenguaje Ladder aplicado al PLC.
- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo y el primer nivel de presión, utilizando el lenguaje C++ aplicado al Microcontrolador.
- Programar el accionamiento de los actuadores 1 y 2 utilizando el lenguaje Ladder aplicado al PLC.
- Programar el accionamiento de los actuadores 1 y 2 utilizando el lenguaje C++ aplicado al Microcontrolador.

8.7.2 Fundamentación previa

- Revisar manual teórico.
- Identificar las principales características de los equipos utilizados en la elaboración de la práctica.

8.7.3 Procedimientos de seguridad

Antes de la elaboración de las prácticas se deben tener previo conocimiento de los procedimientos de seguridad que se deben seguir, estos se exponen en el inicio de este manual.

8.7.4 Ubicación e identificación de los componentes. Identifique cada uno de los componentes mencionados en la tabla 19. Para ver la distribución de los elementos ubicados en el panel ver la figura 135.

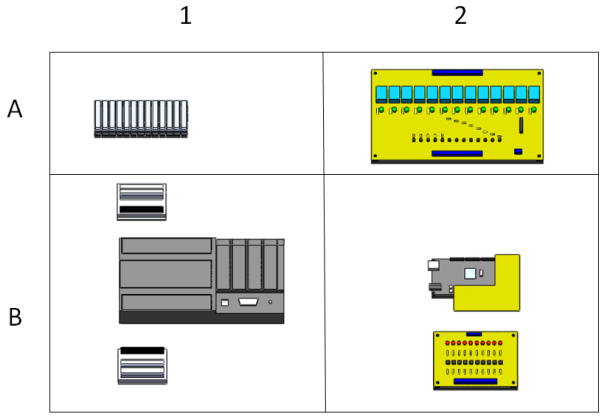
Tabla 19. Componentes requeridos para la elaboración de la práctica tres.

ELEMENTO	NOMENCLATURA	UBICACIÓN
Parada de emergencia	-S20	Tablero de mando
Selector de energía principal	-S21	Tablero de mando
Indicador de energía principal	-H1	Tablero de mando
Selector de equipo (PLC o Microcontrolador)	-S22	Tablero de mando
Indicador del PLC	-H5	Tablero de mando
Indicador de Microncontrolador	-H6	Tablero de mando
Pulsador inicio del motor	-S10	Tablero de mando
Pulsador apagado del motor	-S11	Tablero de mando
Indicador motor encendido	-H2	Tablero de mando
Indicador de válvula de presión 1	-H3	Tablero de mando
PLC		B1
Conectores de entradas al PLC	C1	B1
Conectores de salidas del PLC	C2	B1
Microcontrolador		B2
Tarjeta de interface de entadas		B1
Tarjeta de potencia		A2
Bornes portafusibles	-F	A1
Microcontrolador		B2
Contactores arranque del motor	-KM1, -KM2, -KM3	
Relevos para arranque del motor	-K10, -K11 -KA10, -KA11	Tarjeta de potencia Tablero de potencia
Relevo para activación de 1^{er} nivel de presión	-K12	Tarjeta de potencia
Relevo para activación solenoide V1-0 de la válvula 1	-k0	Tarjeta de potencia
Relevo para activación solenoide V1-1 de la válvula 1	-k1	Tarjeta de potencia
Relevo para activación solenoide V2-2 de la válvula 2	-k2	Tarjeta de potencia
Relevo para activación solenoide V2-3 de la válvula 2	-k3	Tarjeta de potencia
Sensor 0	-S0	Inicio carrera del actuador 1
Sensor 1	-S1	Final de carrera del actuador 1

Fuente: Autores.

Sensor 2	-S3	Inicio carrera del actuador 2
Sensor 3	-S2	Final de carrera del actuador 2
Válvula primer nivel de presión	-S12	Unidad hidráulica
Válvula 1	V1	Unidad hidráulica
Solenoides válvula 1	V1-0 (sale actuador) V1-1 (entra actuador)	Unidad hidráulica
Válvula 2	V2	Unidad hidráulica
Solenoides válvula 2	V2-2 (sale actuador) V2-3 (entra actuador)	Unidad hidráulica
Actuador 1		Unidad hidráulica
Actuador 2		Unidad hidráulica
Válvula de 1er nivel de presión	V12	Unidad hidráulica
Alarma	H9	Tablero eléctrico

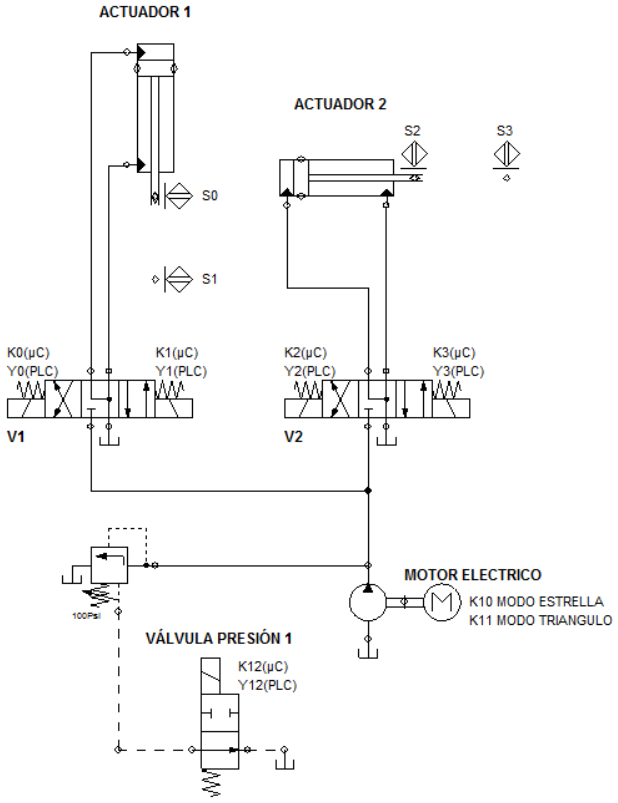
Figura 135. Distribución del los elementos en el panel.



Fuente: Autores.

8.7.5 Reconocimiento del circuito hidráulico. El circuito hidráulico a controlar consta de un motor eléctrico, una electroválvula de presión y dos actuadores controlados cada uno por una electroválvula. La figura 136 ilustra el circuito hidráulico.

Figura 136. Circuito hidráulico de la tercera práctica.



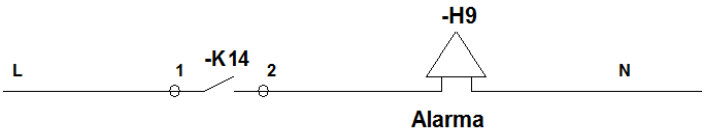
Fuente: Autores.

8.7.6 Reconocimiento de los circuitos eléctricos. Analice y discuta cada uno de los circuitos eléctricos requeridos para la elaboración de la práctica.

Ver figura 119, 120, 121, 122.

La figura 137 ilustra el esquema de accionamiento de la alarma, la cual debe sonar en caso de no haber señal por parte de alguno de los sensores.

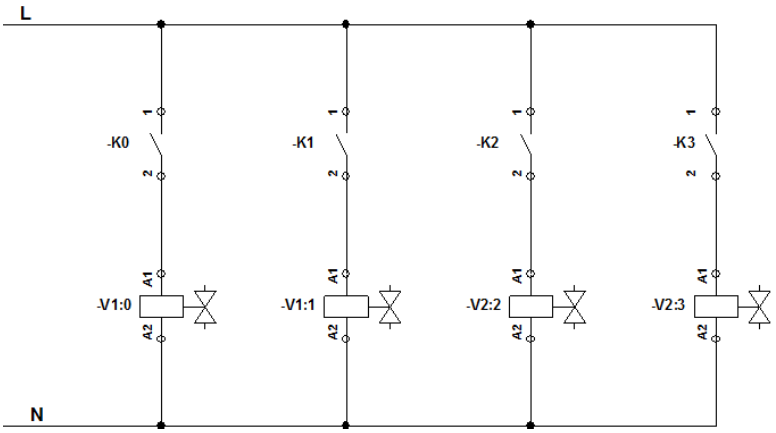
Figura 137. Esquema de accionamiento de la alarma.



Fuente Autores.

La figura 138 muestra el esquema básico de las conexiones eléctricas de las electroválvulas 1 y 2.

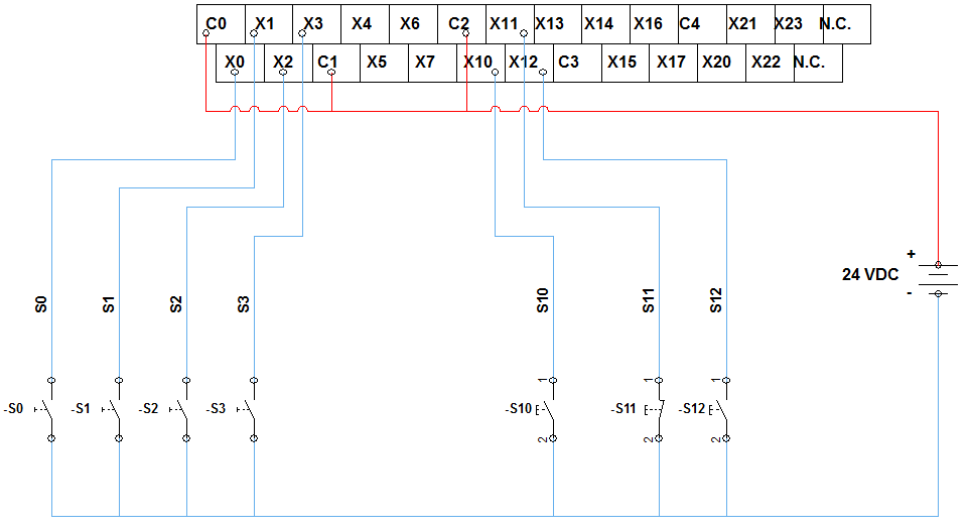
Figura 138. Esquema eléctrico del accionamiento de las electroválvulas 1 y 2.



Fuente: Autores.

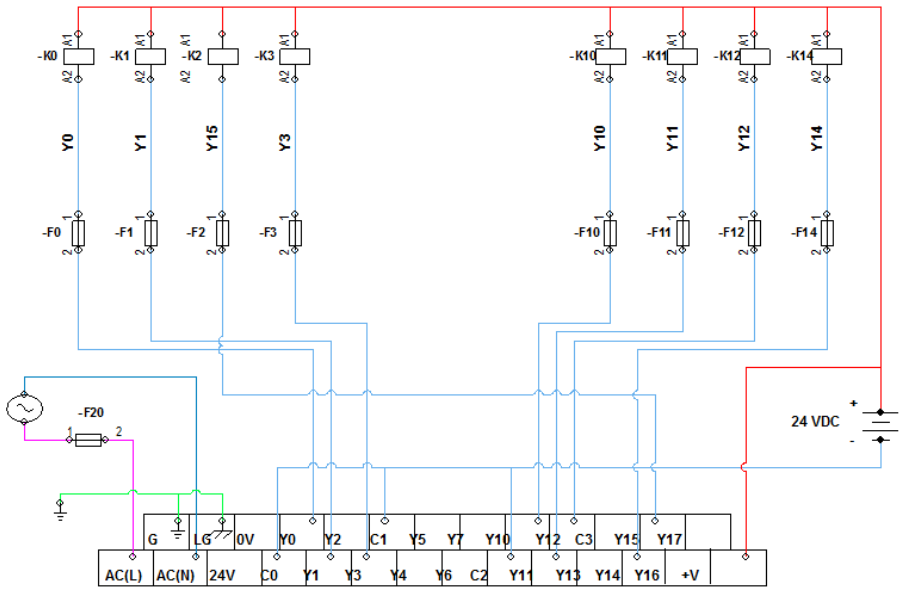
En la figura139 y 140 observan las conexiones de entradas y salidas respectivamente del PLC que se requieren para la elaboración de la práctica 3.

Figura 139. Esquema eléctrico de las conexiones de entradas del PLC requeridas para la tercera práctica.



Fuente: Autores.

Figura 140. Esquema eléctrico de las conexiones de salida del PLC requeridas para la tercera práctica.



Fuente: Autores.

La tabla 20 muestra los puertos de salidas del Microncontrolador y los relevos que activan en la tarjeta de potencia, para la elaboración de la práctica.

Tabla 20. Puertos de salidas de Microcontrolador y los relevos que activan en la tarjeta de potencia (Práctica tres).

PUERTOS DIGITALES DE LA TARJETA ARDUINO	SEÑALES DE SALIDA- TARJETA DE POTENCIA
27	K14
31	K12
33	K11
35	K10
45	K3
47	K2
49	K1
51	K0

Fuente: Autores.

La tabla 21 muestra los puertos de entrada del Microncontrolador y las señales de los sensores que llegan a estos puertos.

Tabla 21. Puertos de entrada del Microncontrolador y señales de los sensores que llegan a los mismos (Práctica tres).

PUERTOS ANALÓGICOS DE LATARJETA ARDUINO	SEÑALES DE ENTRADA-TARJETA INTERFAZ
A0	S2
A1	S3
A6	S10
A7	S11
A8	S0
A9	S1

Fuente: Autores.

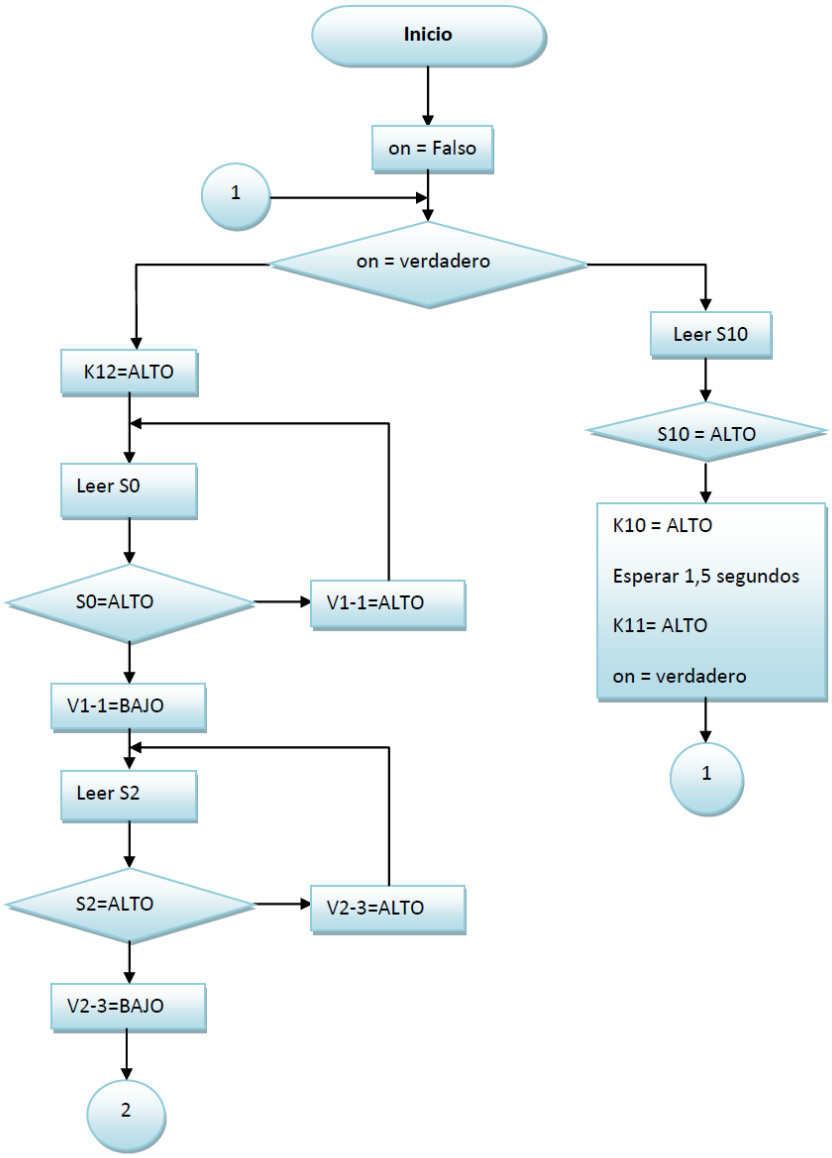
8.7.7 Programación. Programe la siguiente secuencia para el PLC y el Microncontrolador.

a) Al presionar S10 se enciende motor por medio del relevo K10.
b) 1.5 seg después encender el relevo K11 para pasar de estrella a triangulo.
c) 1 seg después active el primer nivel de presión por medio del relevo K12.
d) Compruebe que los actuadores 1 y 2 se encuentren en posición correcta.
e) Realice la siguiente secuencia:
    1. Active solenoide V1-0 para sacar el actuador 1
    2. Active solenoide V2-2 para sacar el actuador 2.
    3. Active solenoide V1-1 para devolver a su posición inicial el actuador 1.
    4. Active solenoide V2-3 para devolver a su posición inicial el actuador 2.
    5. Desactive 1er nivel de presión.
    6. Apague motor.
f) Programe la alarma de tal forma que se active, y al mismo tiempo se apague le motor de modo seguro, si los sensores no envían señales. Tenga en cuenta

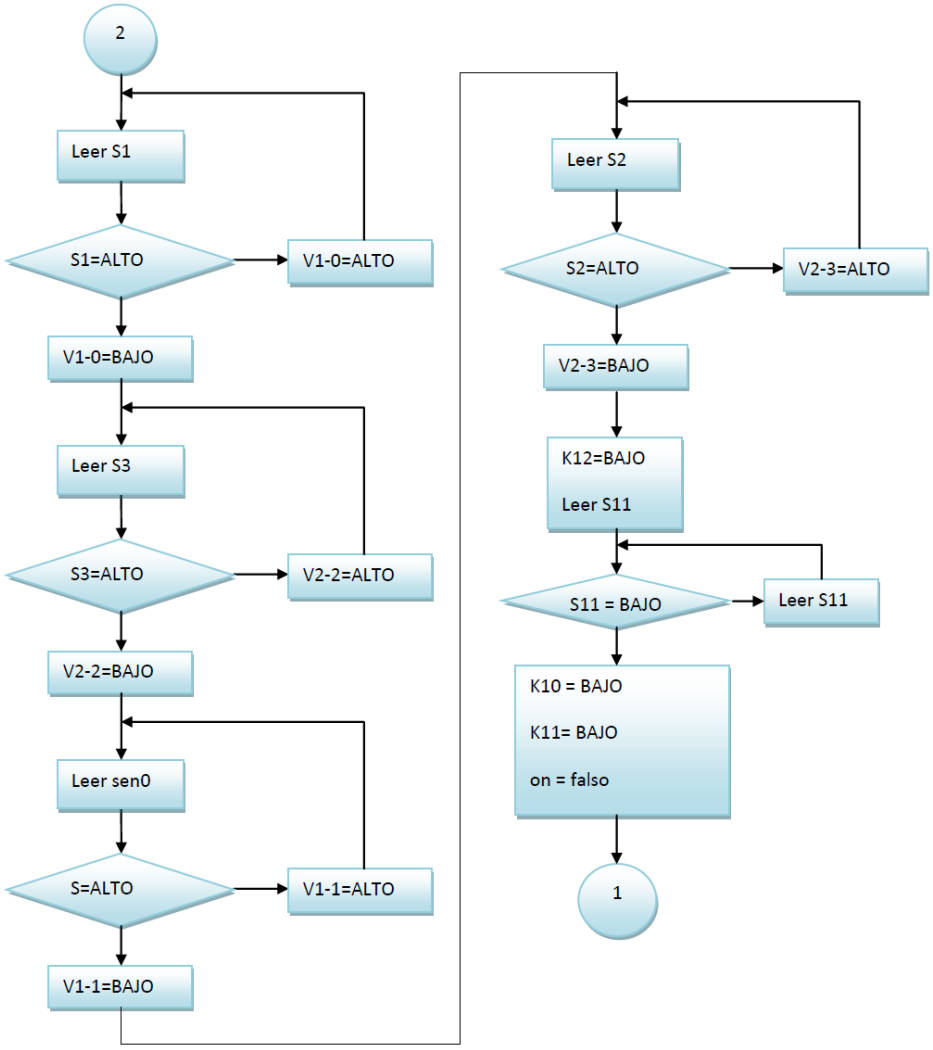
que el tiempo que se demora el actuador uno en ir de un extremo a otro es de 6 segundos, y el del actuador dos es de 3 segundos.

En la figura 141 se muestra un ejemplo del diagrama de flujo del programa para la tercera práctica.

Figura 141. Diagrama de flujo del programa de tercera práctica.



Fuente: Autores.



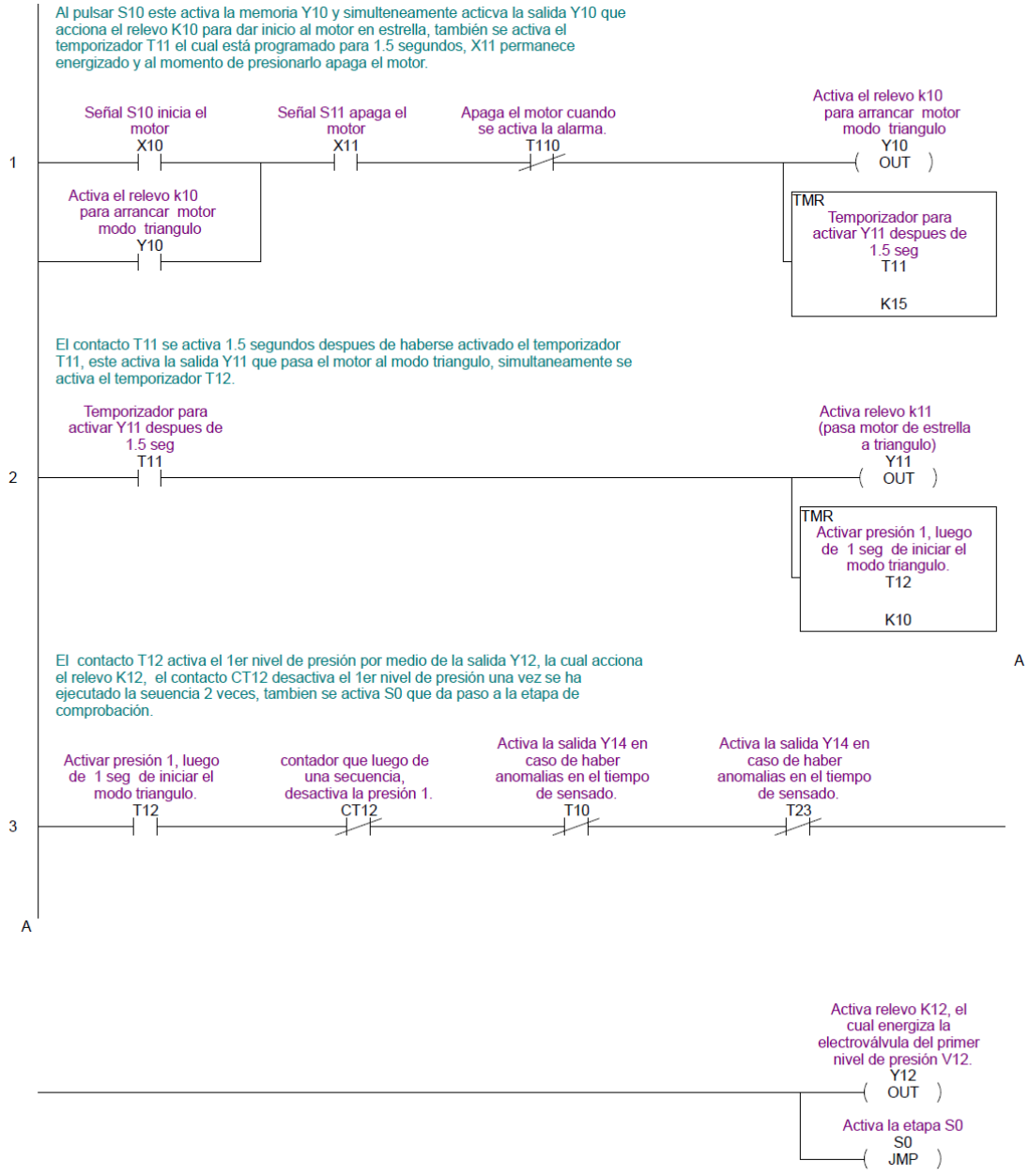
A. PLC

- Antes de realizar el programa lea la guía de programación para el PLC.
- Utilice el software DirectSOFT 5 para elaborar el programa.

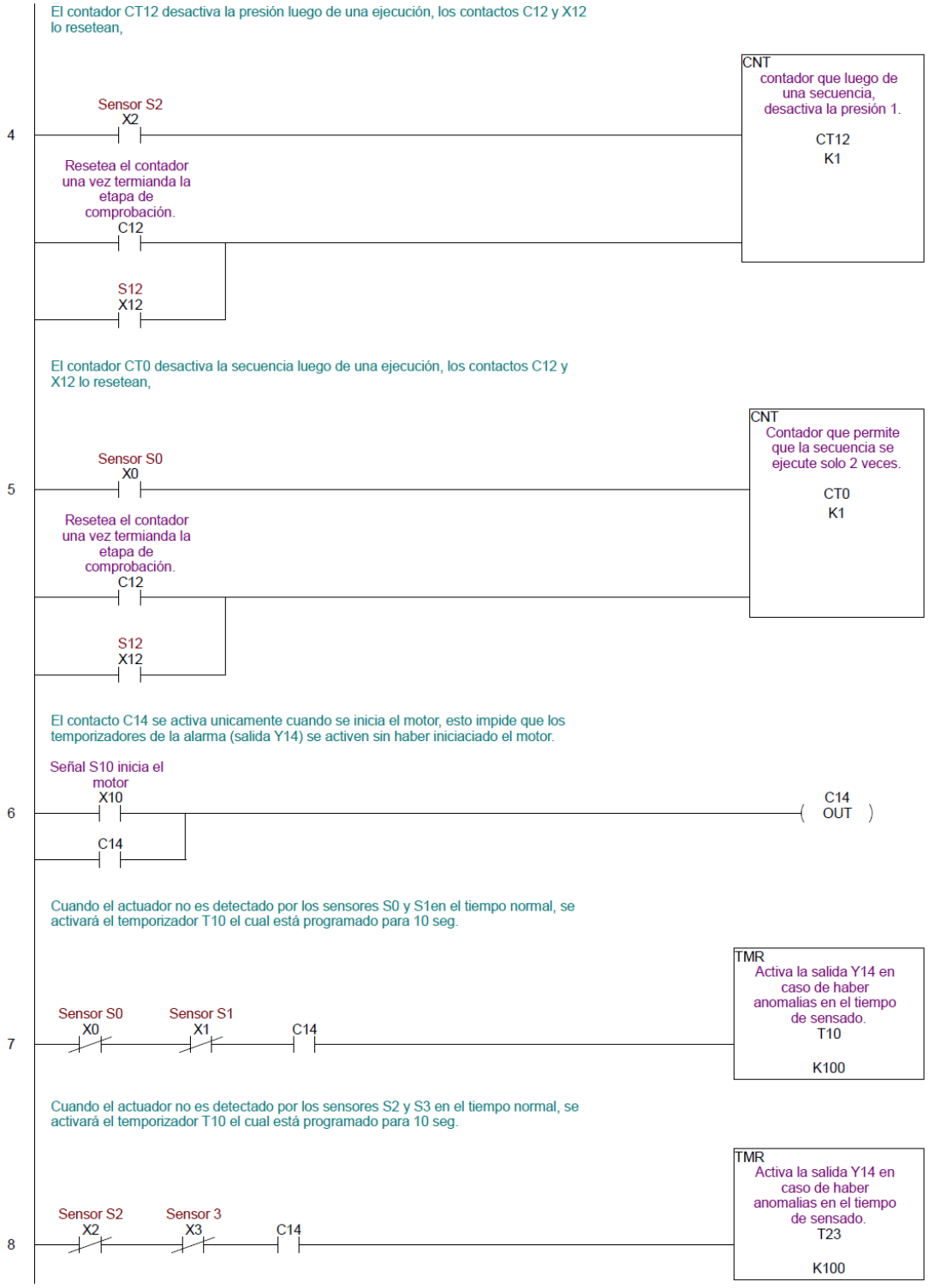
Ejemplo de programación

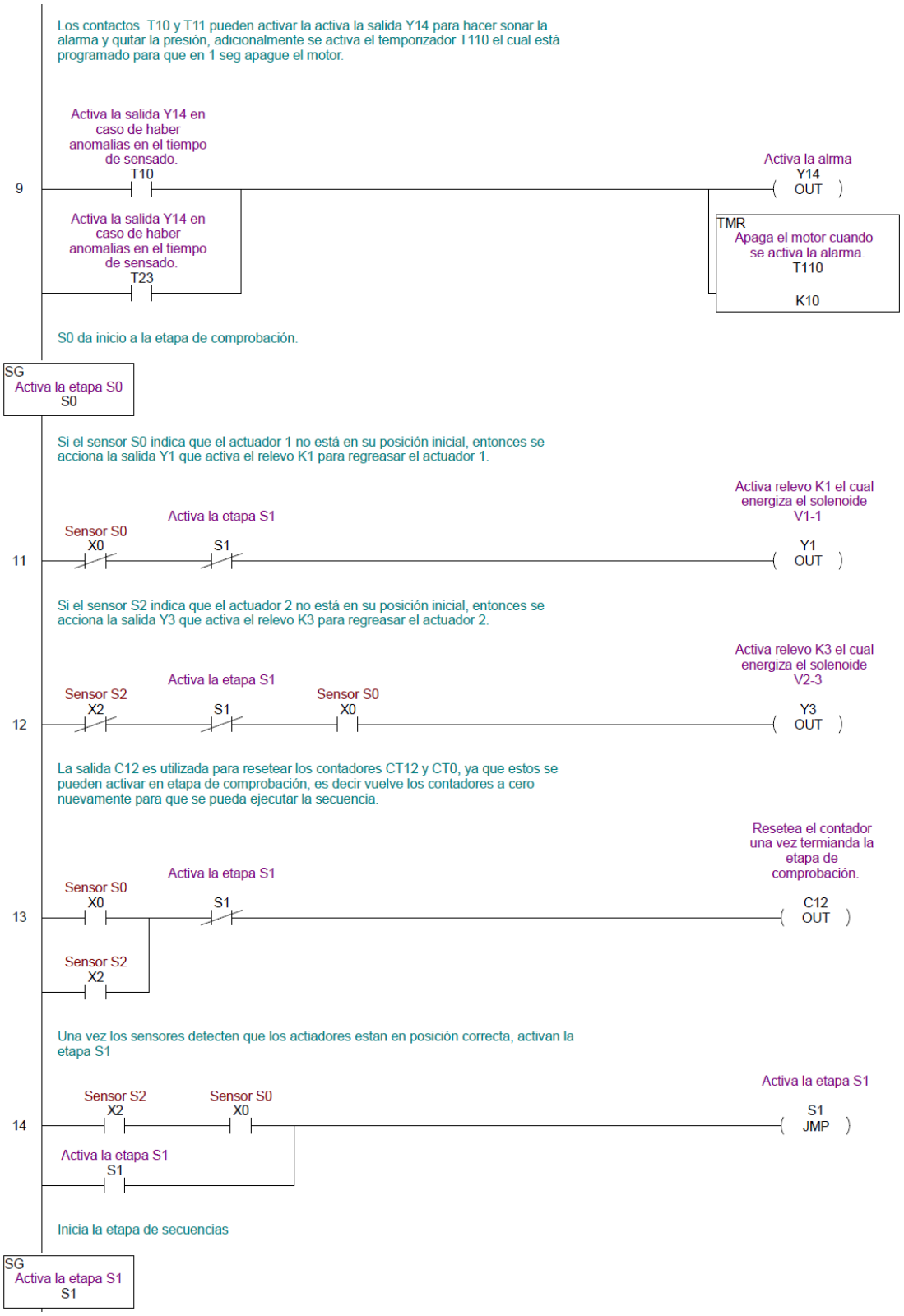
En la figura 142 se muestra un ejemplo de programación para el PLC de la tercera práctica.

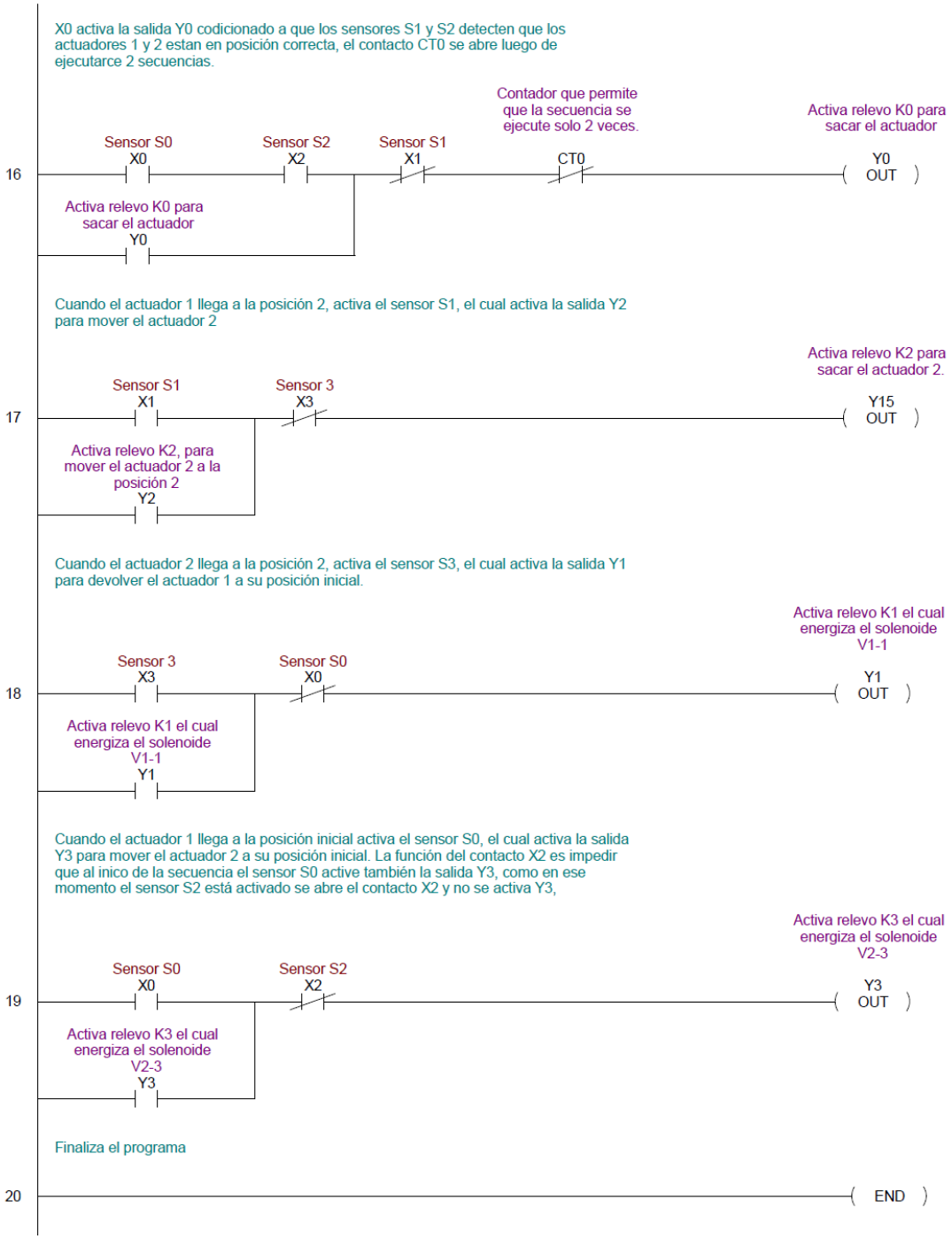
Figura 142. Ejemplo de programación del PLC de la tercera práctica.



Fuente: Autores.


El contador CT12 desactiva la presión luego de una ejecución, los contactos C12 y X12 lo resetean,
CNT
contador que luego de una secuencia, desactiva la presión 1.
CT12
K1
4
Sensor S2
X2
Resetea el contador una vez termianda la etapa de comprobación.
C12
S12
X12
El contador CT0 desactiva la secuencia luego de una ejecución, los contactos C12 y X12 lo resetean,
CNT
Contador que permite que la secuencia se ejecute solo 2 veces.
CT0
K1
5
Sensor S0
X0
Resetea el contador una vez termianda la etapa de comprobación.
C12
S12
X12
El contacto C14 se activa unicamente cuando se inicia el motor, esto impide que los temporizadores de la alarma (salida Y14) se activen sin haber iniciaciado el motor.
Señal S10 inicia el motor
X10
C14
6
C14
OUT
Cuando el actuador no es detectado por los sensores S0 y S1en el tiempo normal, se activará el temporizador T10 el cual está programado para 10 seg.
TMR
Activa la salida Y14 en caso de haber anomalias en el tiempo de sensado.
T10
K100
7
Sensor S0
X0
Sensor S1
X1
C14
Cuando el actuador no es detectado por los sensores S2 y S3 en el tiempo normal, se activará el temporizador T10 el cual está programado para 10 seg.
TMR
Activa la salida Y14 en caso de haber anomalias en el tiempo de sensado.
T23
K100
8
Sensor S2
X2
Sensor 3
X3
C14


Los contactos T10 y T11 pueden activar la activa la salida Y14 para hacer sonar la alarma y quitar la presión, adicionalmente se activa el temporizador T110 el cual está programado para que en 1 seg apague el motor.
Activa la salida Y14 en caso de haber anomalias en el tiempo de sensado.
T10
Activa la alrma
Y14
OUT
9
Activa la salida Y14 en caso de haber anomalias en el tiempo de sensado.
T23
TMR
Apaga el motor cuando se activa la alarma.
T110
K10
S0 da inicio a la etapa de comprobación.
SG
Activa la etapa S0
S0
Si el sensor S0 indica que el actuador 1 no está en su posición inicial, entonces se acciona la salida Y1 que activa el relevo K1 para regreasar el actuador 1.
Activa relevo K1 el cual energiza el solenoide V1-1
Activa la etapa S1
Sensor S0
X0
S1
Y1
OUT
11
Si el sensor S2 indica que el actuador 2 no está en su posición inicial, entonces se acciona la salida Y3 que activa el relevo K3 para regreasar el actuador 2.
Activa relevo K3 el cual energiza el solenoide V2-3
Activa la etapa S1
Sensor S2
X2
S1
Sensor S0
X0
Y3
OUT
12
La salida C12 es utilizada para resetear los contadores CT12 y CT0, ya que estos se pueden activar en etapa de comprobación, es decir vuelve los contadores a cero nuevamente para que se pueda ejecutar la secuencia.
Resetea el contador una vez termianda la etapa de comprobación.
Activa la etapa S1
Sensor S0
X0
S1
C12
OUT
13
Sensor S2
X2
Una vez los sensores detecten que los actiadores estan en posición correcta, activan la etapa S1
Activa la etapa S1
Sensor S2
X2
Sensor S0
X0
S1
JMP
14
Activa la etapa S1
S1
Inicia la etapa de secuencias
SG
Activa la etapa S1
S1


X0 activa la salida Y0 codicionado a que los sensores S1 y S2 detecten que los actuadores 1 y 2 estan en posición correcta, el contacto CT0 se abre luego de ejecutarce 2 secuencias.
Contador que permite que la secuencia se ejecute solo 2 veces.
Activa relevo K0 para sacar el actuador
Sensor S0
X0
Sensor S2
X2
Sensor S1
X1
CT0
Y0
OUT
16
Activa relevo K0 para sacar el actuador
Y0
Cuando el actuador 1 llega a la posición 2, activa el sensor S1, el cual activa la salida Y2 para mover el actuador 2
Activa relevo K2 para sacar el actuador 2.
Sensor S1
X1
Sensor 3
X3
Y15
OUT
17
Activa relevo K2, para mover el actuador 2 a la posición 2
Y2
Cuando el actuador 2 llega a la posición 2, activa el sensor S3, el cual activa la salida Y1 para devolver el actuador 1 a su posición inicial.
Activa relevo K1 el cual energiza el solenoide V1-1
Sensor 3
X3
Sensor S0
X0
Y1
OUT
18
Activa relevo K1 el cual energiza el solenoide V1-1
Y1
Cuando el actuador 1 llega a la posición inicial activa el sensor S0, el cual activa la salida Y3 para mover el actuador 2 a su posición inicial. La función del contacto X2 es impedir que al inico de la secuencia el sensor S0 active también la salida Y3, como en ese momento el sensor S2 está activado se abre el contacto X2 y no se activa Y3,
Activa relevo K3 el cual energiza el solenoide V2-3
Sensor S0
X0
Sensor S2
X2
Y3
OUT
19
Activa relevo K3 el cual energiza el solenoide V2-3
Y3
Finaliza el programa
20
END

B. Microcontrolador

- Antes de realizar el programa lea la guía de programación para el Microcontrolador.
- Utilice el software Arduino para elaborar el programa.

Ejemplo de programación.

```
/*  Ejemplo  tercera practica
 Cilindro número 1 ( + ),  Cilindro número 2 ( + ),
 Cilindro número 1 ( - ),  Cilindro número 2 ( - ),
 NOTA:  *digitar el número de ciclos(cic) o cero(0) para repetir ciclos
indefinidamente
 *digitar los tiempos de retraso (t1mas, t1men, t2mas, t2men)
 */
int cic = 3;
int t1mas = 2;     //tiempo de permanencia cilindro 1 +(segundos)
int t1men = 0;     //tiempo de permanencia cilindro 1 -(segundos)
int t2mas = 0;     //tiempo de permanencia cilindro 2 +(segundos)
int t2men = 0;     //tiempo de permanencia cilindro 2 -(segundos)
//variables
int var1=0;         //variables de lectura  de la señal S10 (encendido de motor)
int var2=0;
int var3=0;      //variables de lectura  de la señal S11 (apagar motor)
int var4=0;

int S10 = A6;     // pulsador encendido entrada analógica  6
int S11 = A7;     // pulsador apagado entrada analógica  7
int senal_ens=0;    // almacena  señal encendido
int senal_apag=1;  // almacena  señal apagado
int on = 0;           // estado del motor
int K10 = 35;     // relevo del encendido modo estrella pin 35
int K11 = 33;     // relevo del apagado modo triangulo pin 33
int K12 = 31;       // 1er nivel de presión pin 31
int K14 = 27;      // alarma
int cont1=0;   // contador de ciclos
int cont2=0;   // contador para actuadores en posición inicial
int cont3=0;   //contador para impresión de número de ciclos
int Tref;    // tiempo de referencia para la alarma

// Cilindro 1
int S0 = A8;     //sensor 0 cilindro 1 ... entrada analógica pin 8
int S1 = A9;     //sensor cilindro 1 ... entrada analógica pin 9
int K0 = 51;     //válvula cilindro 1 +
int K1 = 49;     //válvula cilindro 1 -
int lS0 = 0;
int lS1 = 0;

// cilindro 2
int S2 = A0;     //sensor 2 cilindro 2 ... entrada analógica pin 2
int S3 = A1;     //sensor 3 cilindro 2 ... entrada analógica pin 3
int K2 = 47;     //válvula cilindro 1 +
int K3 = 45;     //válvula cilindro 1 -
int lS2 = 0;
int lS3 = 0;
void setup(){
```

```
  Serial.begin(9600);       // comunicación

  pinMode(K10, OUTPUT);   //define como salida estrella
  pinMode(K11, OUTPUT);   //define como salida triangulo

  pinMode(K12, OUTPUT);   //define como salida presión 1

  pinMode(K14, OUTPUT);   //define como salida la alarma

  //cilindro 1
  pinMode(K0, OUTPUT);   //define como salida electroválvula cil 1 +
  pinMode(K1, OUTPUT);   //define como salida electroválvula cil 1 -

  // cilindro 2
  pinMode(K2, OUTPUT);   //define como salida electroválvula cil 2 +
  pinMode(K3, OUTPUT);   //define como salida electroválvula cil 2 -

}
void loop(){
  encender();
  practica3();
  apagar();
}
void encender(){
  if(on == 0){           // si el motor está apagado entra
    var1 = analogRead(S10);  // lee estado de señal y almacena en var1
    if (var1 >= 500){      // convierte señal en cero o uno
      var1=1;
    }
    else {
      var1=0;
    }
    delay(10);             //retraso de 10 milisegundos
    var2 = analogRead(S10);  // lee estado de señal y almacena en var2
    if (var2 >= 500){      // convierte señal en cero o uno
      var2=1;
    }
    else {
      var2=0;
    }
    if (var1 == var2){      // evita rebote de contacto
      if (var1 != senal_ens){        // selecciona cuando hay  cambio de la señal
        if (var1 == LOW){    // selecciona cuando se suelta el pulsador
        digitalWrite(K14, LOW);
          digitalWrite(K10, HIGH);      //prende el motor modo estrella
          Serial.println("motor prendido modo estrella");   //imprime "motor
prendido modo estrella"
          delay(1500);       // espera 1,5 segundos
          digitalWrite(K11, HIGH);     //   pasa a modo triangulo
          Serial.println("motor prendido modo triangulo");   //imprime "motor
prendido modo triangulo"
          on = 1;
          delay(1000);     // retraso de un segundo
          cont1 = 0;
          cont2 = 0;
          cont3 = 0;
        }
        senal_ens = var1;
      }
    }
  }
```

```
}
void practica3(){
  if (cont2 < 1 && on == 1){
    digitalWrite(K12, HIGH);
    delay(700);
    Serial.println("nivel de presión 1");
    posicioni1();
    posicioni2();
    ciclos();
    digitalWrite(K12, LOW);
    if (on == 1){
      Serial.println("fin de la tarea");
      digitalWrite(K12, LOW);
      Serial.println("venteo");
    }
    cont2++;
  }
}
void posicioni1(){  // función para posicionar cilindro 1
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 1");
  }
}
void posicioni2(){  // función para posicionar cilindro 2
  lS2 = analogRead(S2);
  Tref = millis();
  while (lS2 <= 500 && on == 1){
    digitalWrite(K3, HIGH);
    lS2 = analogRead(S2);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K3, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 2");
  }
}
void ciclos(){   //función para el numero de ciclos
  while(cont1 < cic && on ==1){   // siempre que el contador de ciclos sea menor
que la cantidad de ciclos requeridos hace...
    cont3++;     // suma una unidad a el contador de ciclos para la impresión
    Serial.print("ciclo número ");
    Serial.println(cont3);        //   imprime en número de ciclo
    act_1_mas();
    act_2_mas();
    act_1_men();
    act_2_men();
    if(cic != 0){ // si el número de ciclos es cero el ciclo se repetirá hasta
que se apague el motor
      cont1++;   //suma una unidad al número de ciclos
    }
  }
```

```
}
void act_1_mas(){   // función para cilindro 1( + )
  lS1 = analogRead(S1);
  Tref = millis();
  while (lS1 <= 500 && on == 1){
    digitalWrite(K0, HIGH);
    lS1 = analogRead(S1);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K0, LOW);
  fdelay(t1mas);
}
void act_1_men(){   // función para cilindro 1( - )
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  fdelay(t1men);
}
void act_2_mas(){   // función para cilindro 2( + )
  lS3 = analogRead(S3);
  Tref = millis();
  while (lS3 <= 500 && on == 1){
    digitalWrite(K2, HIGH);
    lS3 = analogRead(S3);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K2, LOW);
  fdelay(t2mas);
}

void act_2_men(){   // función para cilindro 2( - )
  lS2 = analogRead(S2);
  Tref = millis();
  while (lS2 <= 500 && on == 1){
    digitalWrite(K3, HIGH);
    lS2 = analogRead(S2);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K3, LOW);
  fdelay(t2men);
}
void apagar(){
  var3 = analogRead(S11);  // lee estado de señal y almacena en var3
  if (var3 >= 500){      // convierte señal en cero o uno
    var3=1;
  }
  else {
    var3=0;
  }
  delay(10);             //retraso de 10 milisegundos
  var4 = analogRead(S11);  // lee estado de señal y almacena en var4
  if (var4 >= 500){      // convierte señal en cero o uno
```

```
    var4=1;
  }
  else {
    var4=0;
  }
  if (var3 == var4){      // evita rebote de contacto
    if (var3 != senal_apag){        // selecciona cuando hay  cambio de la señal
      if (var3 == HIGH){
        digitalWrite(K14, LOW);
        if (on == 1){    // selecciona cuando se suelta el pulsador
          digitalWrite(K12, LOW);    // ventear
          digitalWrite(K10, LOW);    // apaga motor
          digitalWrite(K11, LOW);
          Serial.println("motor apagado");   //imprime "motor apagado"
          on = 0;
        }
      }
      senal_apag = var3;
    }
  }
}
void fdelay(int s){
  int a = 0;
  while (a <= s*1000 && on == 1){
    a = a + 100;
    delay (100);
    apagar();
  }
}
void alarma(int Talarma, int act){
  int T = -(Tref - millis());
  if (abs(T) >= Talarma && on == 1){
    digitalWrite(K14, HIGH);
    digitalWrite(K12, LOW);
    delay(1000);
    digitalWrite(K10, LOW);
    digitalWrite(K11, LOW);
    on = 0;
    Serial.print("ALERTA: hay problemas en el actuador ");
    Serial.println(act);
    Serial.println("Se recomienda revisar las electroválvulas y los sensores de
dicho actuador");
  }
}
```

8.7.8 Evaluación para los estudiantes.

1. Elabore un programa utilizando el lenguaje Ladder para el PLC y en C++ para el Microcontrolador que permita realizar la secuencia indicada en el ítem de programación.
2. ¿Cuántas fases tiene el motor eléctrico?
3. ¿Cuántos comunes tienen los puertos de entrada del PLC?
4. ¿Qué tipo de señales se están manejando analógicas o digitales?

8.8 PRÁCTICA 4: ARRANQUE DEL MOTOR Y ACCIONAMIENTO DE LOS ACTUADORES 1,2 y 4

8.8.1 Objetivos

- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo utilizando el lenguaje Ladder aplicado al PLC.
- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo utilizando el lenguaje C++ aplicado al Microcontrolador.
- Programar el accionamiento de los actuadores 1, 2 y 4 utilizando el lenguaje Ladder aplicado al PLC.
- Programar el accionamiento de los actuadores 1, 2 y 4 utilizando el lenguaje C++ aplicado al Microcontrolador.

8.8.2 Fundamentación previa

- Revisar manual teórico.
- Identificar las principales características de los equipos utilizados en la elaboración de la práctica.

8.8.3 Procedimientos de seguridad. Antes de la elaboración de las prácticas se deben tener previo conocimiento de los procedimientos de seguridad que se deben seguir, estos se exponen en el inicio de este manual.

8.8.4 Ubicación e identificación de los componentes. Identifique cada uno de los componentes mencionados en la tabla 22 Para ver la distribución de los elementos ubicados en el panel ver la figura 143.

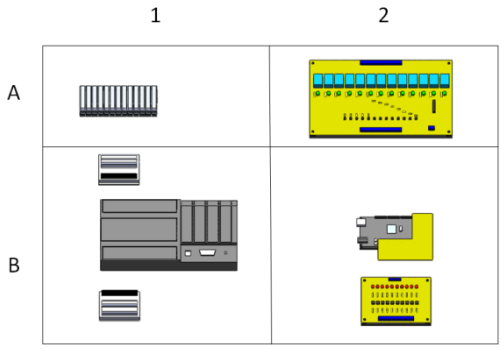
Tabla 22. Componentes requeridos para la elaboración de la práctica cuatro.

ELEMENTO	NOMENCLATURA	UBICACIÓN
Parada de emergencia	-S20	Tablero de mando
Selector de energía principal	-S21	Tablero de mando
Indicador de energía principal	-H1	Tablero de mando
Selector de equipo (PLC o Microcontrolador)	-S22	Tablero de mando
Indicador del PLC	-H5	Tablero de mando
Indicador de Microncontrolador	-H6	Tablero de mando
Pulsador inicio del motor	-S10	Tablero de mando
Pulsador apagado del motor	-S11	Tablero de mando
Indicador motor encendido	-H2	Tablero de mando
Indicador de válvula de presión 1	-H3	Tablero de mando
PLC		B1
Conectores de entradas al PLC	C1	B1
Conectores de salidas del PLC	C2	B1
Microcontrolador		B2
Tarjeta de interface de entadas		B1
Tarjeta de potencia		A2
Bornes portafusibles	-F	A1
Microcontrolador		B2
Contactores arranque del motor	-KM1, -KM2, -KM3	
Relevos para arranque del motor	-K10, -K11 -KA10, -KA11	Tarjeta de potencia Tablero de potencia
Relevo para activación de 1^{er} nivel de presión	-K12	Tarjeta de potencia
Relevo para activación de 2^{er} nivel de presión	-K13	Tarjeta de potencia
Relevo para activación solenoide V1-0 de la válvula 1	-k0	Tarjeta de potencia
Relevo para activación solenoide V1-1 de la válvula 1	-k1	Tarjeta de potencia
Relevo para activación solenoide V2-2 de la válvula 1	-k2	Tarjeta de potencia
Relevo para activación solenoide V2-3 de la válvula 1	-k3	Tarjeta de potencia
Relevo para activación solenoide V4-6 de la válvula 1	-k6	Tarjeta de potencia
Relevo para activación solenoide V4-7 de la válvula 1	-k7	Tarjeta de potencia
Sensor 0	-S0	Inicio de carrera del actuador 1
Sensor 1	-S1	final de carrera del actuador 1
Sensor 2	-S2	Inicio de carrera del actuador 2
Sensor 3	-S3	final de carrera del actuador 2

Fuente: Autores.

Sensor 6	-S6	Inicio de carrera de la carga
Sensor 7	-S7	Final de carrera de la carga
Válvula primer nivel de presión	-S12	Unidad hidráulica
Válvula 1	V1	Unidad hidráulica
Solenoides válvula 1	V1-0 (sale actuador) V1-1 (entra actuador)	Unidad hidráulica
Válvula 2	V2	Unidad hidráulica
Solenoides válvula 2	V2-2 (sale actuador) V2-3 (entra actuador)	Unidad hidráulica
Válvula 4	V4	Unidad hidráulica
Solenoides válvula 4	V4-6 (sube carga) V4-7 (baja carga)	Unidad hidráulica
Actuador 1		Unidad hidráulica
Actuador 2		Unidad hidráulica
Actuador 4		Unidad hidráulica
Válvula 1er nivel de presión	-V12	Unidad hidráulica
Válvula 2do nivel de presión	-V13	Unidad hidráulica

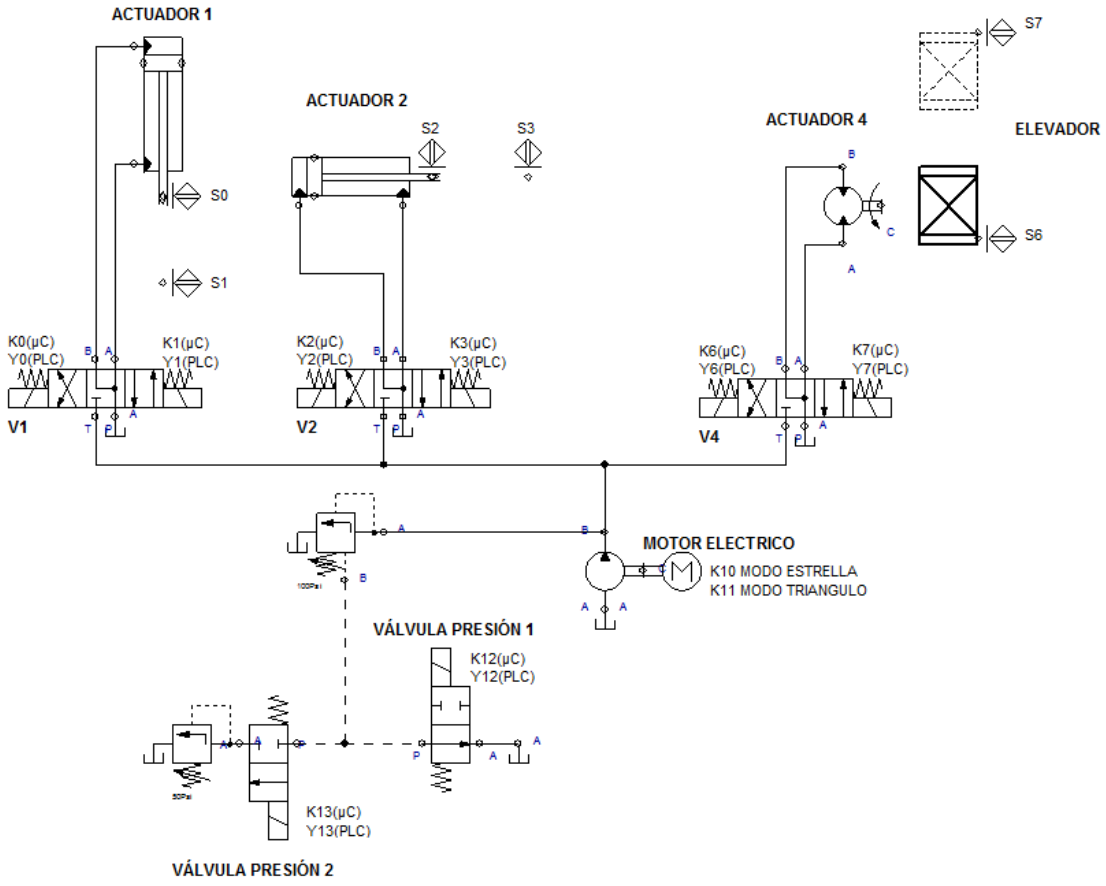
Figura 143. Distribución del los elementos en el panel.



Fuente: Autores.

8.8.5 Reconocimiento del circuito hidráulico. El circuito hidráulico a controlar consta de un motor eléctrico, dos electroválvulas de presión, dos actuadores tipo cilindro de doble efecto controlados cada uno por una electroválvula y un motor hidráulico que acciona un malacate. La figura 144 ilustra el circuito hidráulico.

Figura 144. Circuito hidráulico de la cuarta práctica.



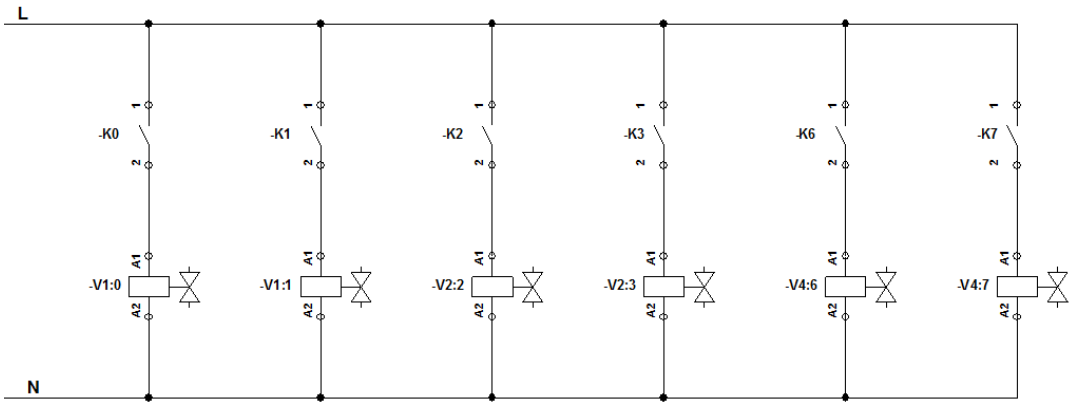
Fuente: Autores.

8.8.6 Reconocimiento de los circuitos eléctricos. Analice y discuta cada uno de los circuitos eléctricos requeridos para la elaboración de la práctica.

Ver figura 119, 120, 121, 122.

La figura 145 muestra el esquema eléctrico para el accionamiento de las electroválvulas 1, 2, 4 requeridas para la práctica 4.

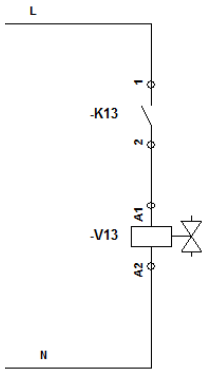
Figura 145. Esquema eléctrico del accionamiento de las electroválvulas 1, 2 y 4.



Fuente: Autores.

Al activar el relvo K13 se energiza el solenoide de la válvula del segundo nivel de presión, tal como se muestra en la figura 146.

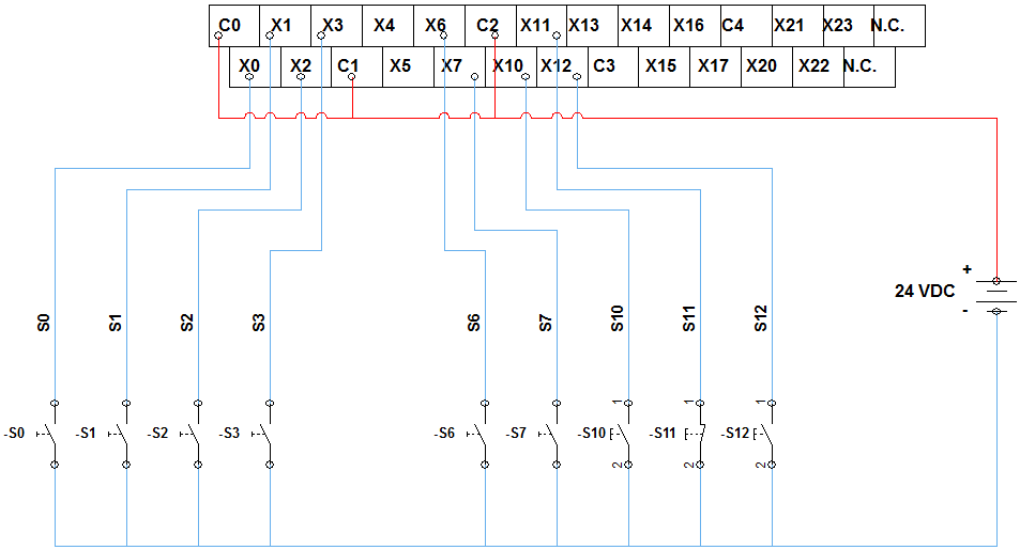
Figura 146. Esquema eléctrico de la válvula del segundo nivel de presión.



Fuente: Autores.

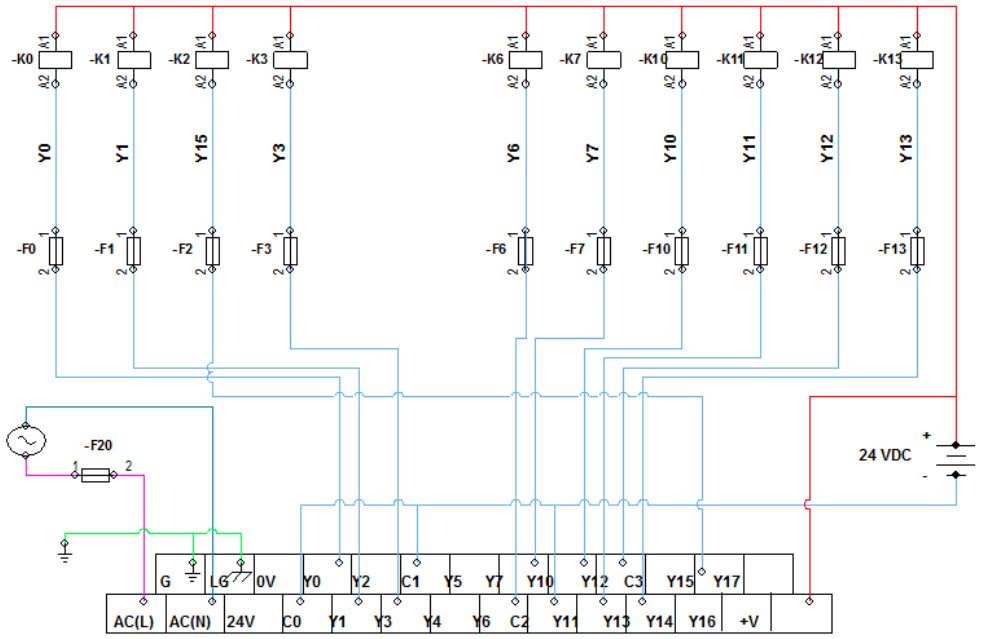
En las figura 147 y 148 se observan las conexiones de entradas y salidas del PLC que se requieren para la elaboración de la práctica 4.

Figura 147. Esquema eléctrico de las conexiones de entrada del PLC requeridas para la práctica cuatro.



Fuente: Autores.

Figura 148. Esquema eléctrico de las conexiones de salida del PLC requeridas para la práctica cuatro.



Fuente: Autores.

La tabla 23 muestra los puertos de salidas del Microncontrolador y los relevos que activan en la tarjeta de potencia, para la elaboración de la práctica.

Tabla 23. Puertos de salidas de Microcontrolador y los relevos que activan en la tarjeta de potencia (Práctica cuatro).

PUERTOS DIGITALES DE LA TARJETA ARDUINO	SEÑALES DE SALIDA-TARJETA DE POTENCIA
29	K13
31	K12
33	K11
35	K10
37	K7
39	K6
45	K3
47	K2
49	K1
51	K0

Fuente: Autores.

La tabla 24 muestra los puertos de entrada del Microncontrolador y las señales de los sensores que llegan a estos puertos.

Tabla 24. Puertos de entrada del Microncontrolador y señales de los sensores que llegan a los mismos (Práctica cuatro).

PUERTOS ANALÓGICOS DE LATARJETA ARDUINO	SEÑALES DE ENTRADA-TARJETA INTERFAZ
A0	S2
A1	S3
A4	S6
A5	S7
A6	S10
A7	S11
A8	S0
A9	S1

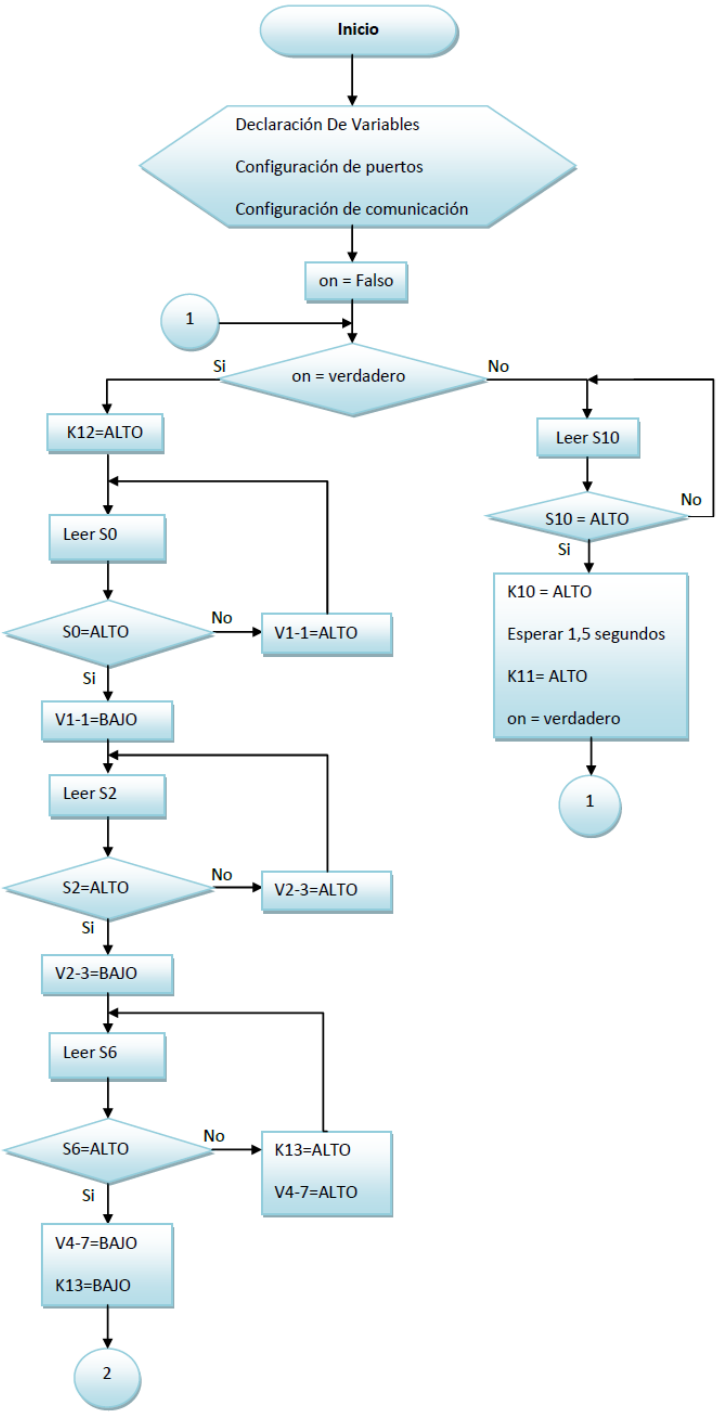
Fuente: Autores.

8.8.7 Programación. Programe la siguiente secuencia para el PLC y el Microncontrolador.

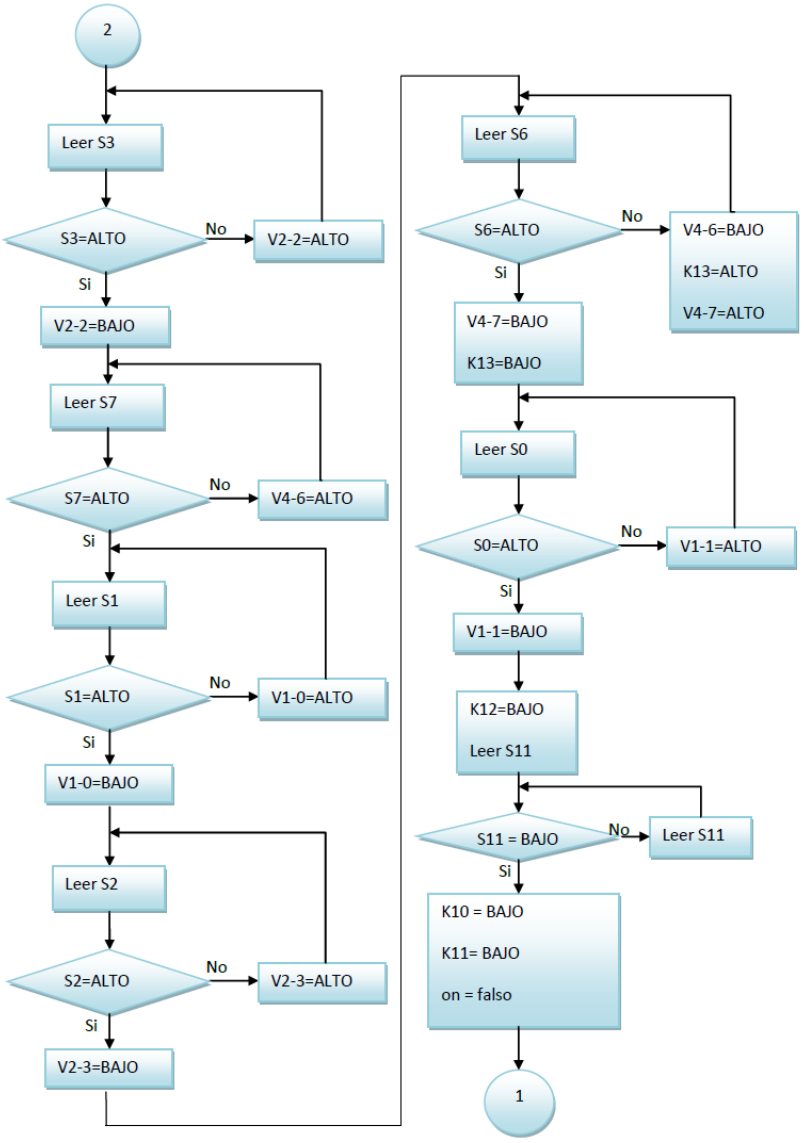
a) Al presionar S10 se enciende motor por medio del relevo K10.
b) 1.5 seg después encender el relevo K11 para pasar de estrella a triangulo.
c) 1 seg después active el primer nivel de presión por medio del relevo K12.
d) Compruebe que los actuadores 1, 2 y 4 se encuentren en posición correcta.
e) Realice la siguiente secuencia:
    1. Active solenoide V2-2 para sacar el actuador 2
    2. Active solenoide V4-6 para subir la carga.
    3. Active solenoide V0-0 para sacar el actuador 1.
    4. Active solenoide V2-3 para devolver a su posición inicial el actuador 2.
    5. Active solenoide V13 para activar 2do nivel de presión.
    6. 0.7 segundos después Active solenoide V4-7 para bajar la carga.
    7. Desactive 2do nivel de presión.
    8. Active solenoide V0-1 para devolver a su posición inicial el actuador 1.
    9. Desactive 1er nivel de presión.
    10. Apague el motor.

En la figura 149 se muestra un ejemplo del diagrama de flujo del programa para la cuarta práctica.

Figura 149. Diagrama de flujo del programa de cuarta práctica.



Fuente. Autores.



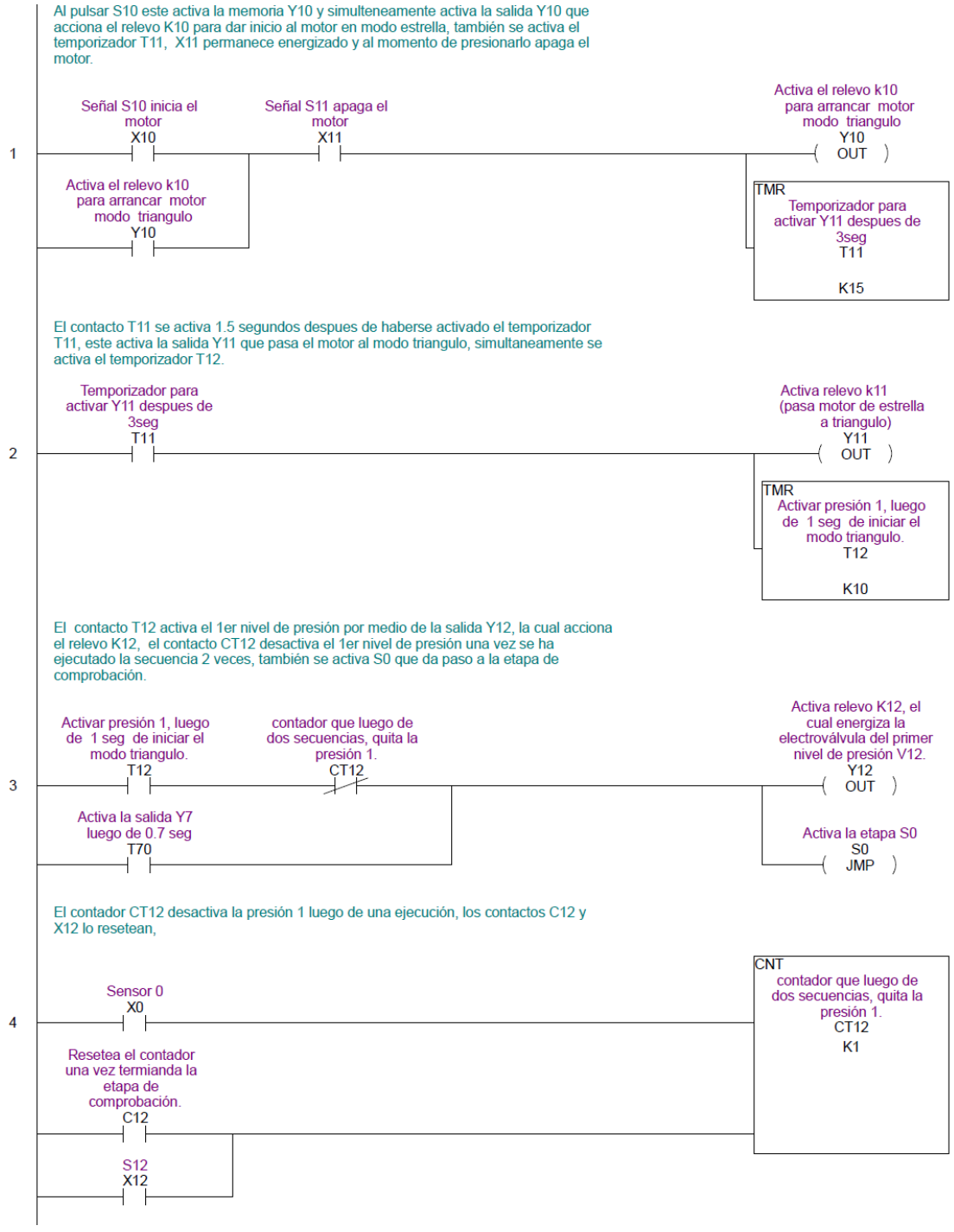
A. PLC

- Antes de realizar el programa lea la guía de programación para el PLC.
- Utilice el software DirectSOFT 5 para elaborar el programa.

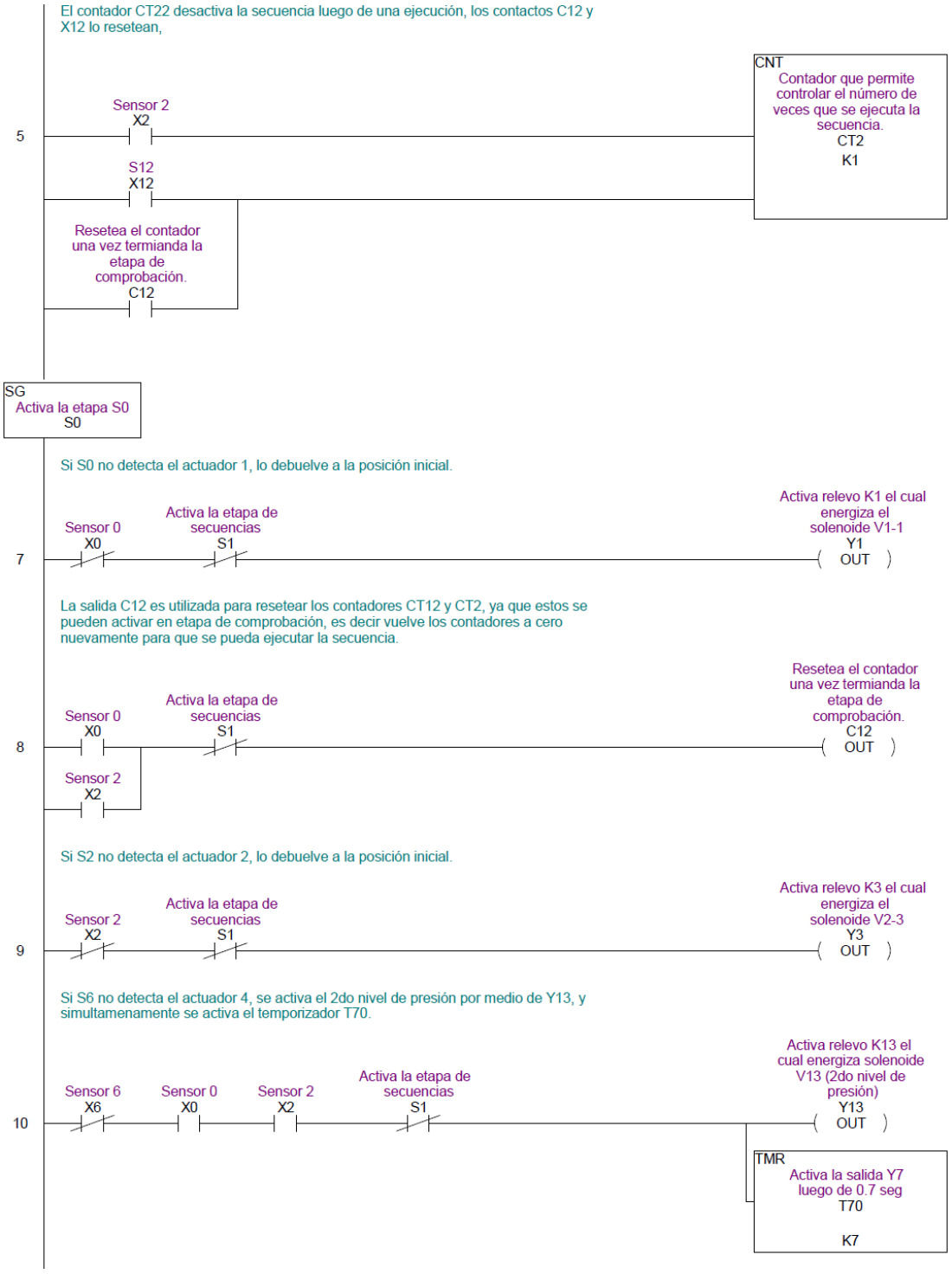
Ejemplo de programación.

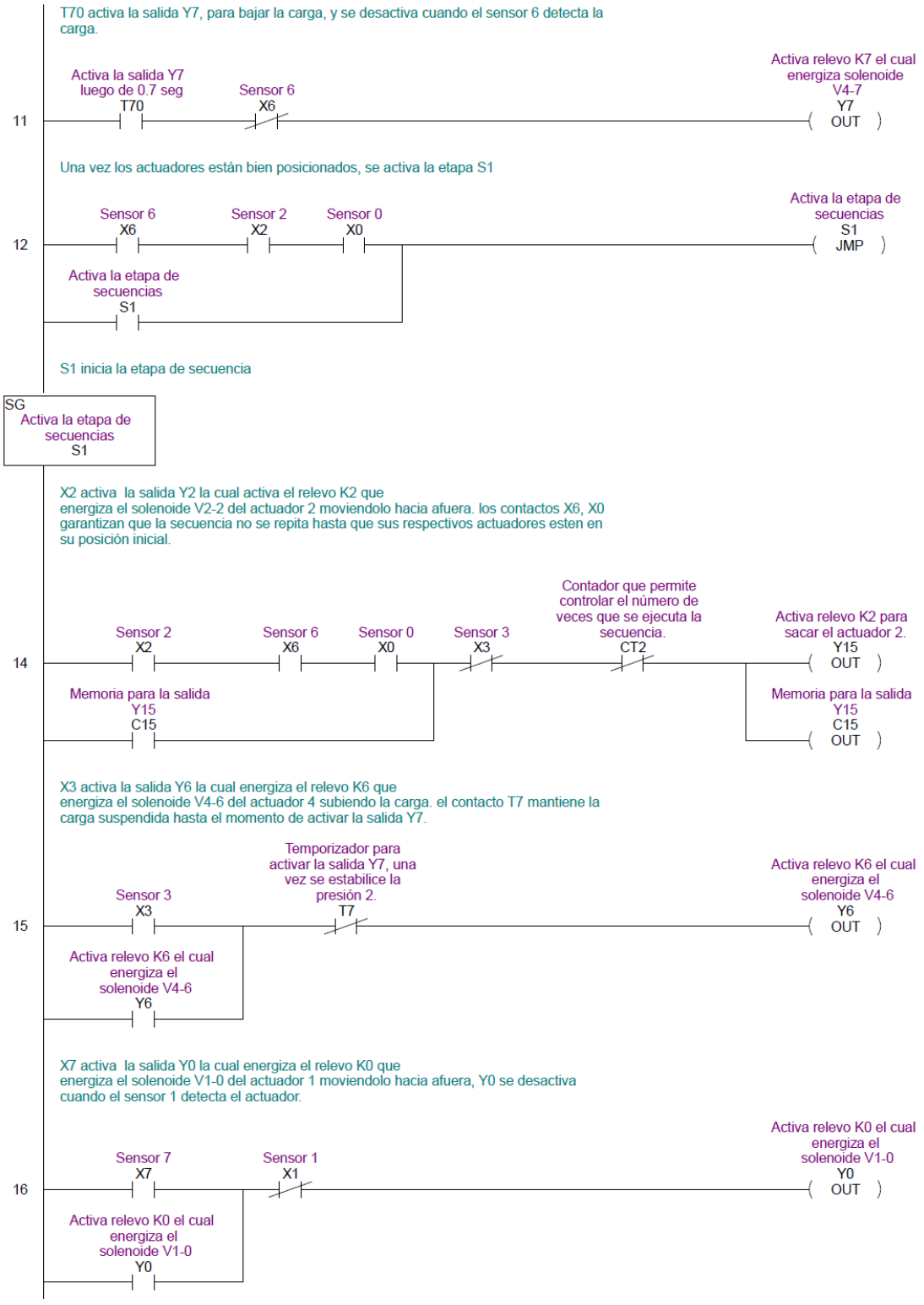
En la figura 150 se muestra un ejemplo de programación para el PLC de la cuarta práctica.

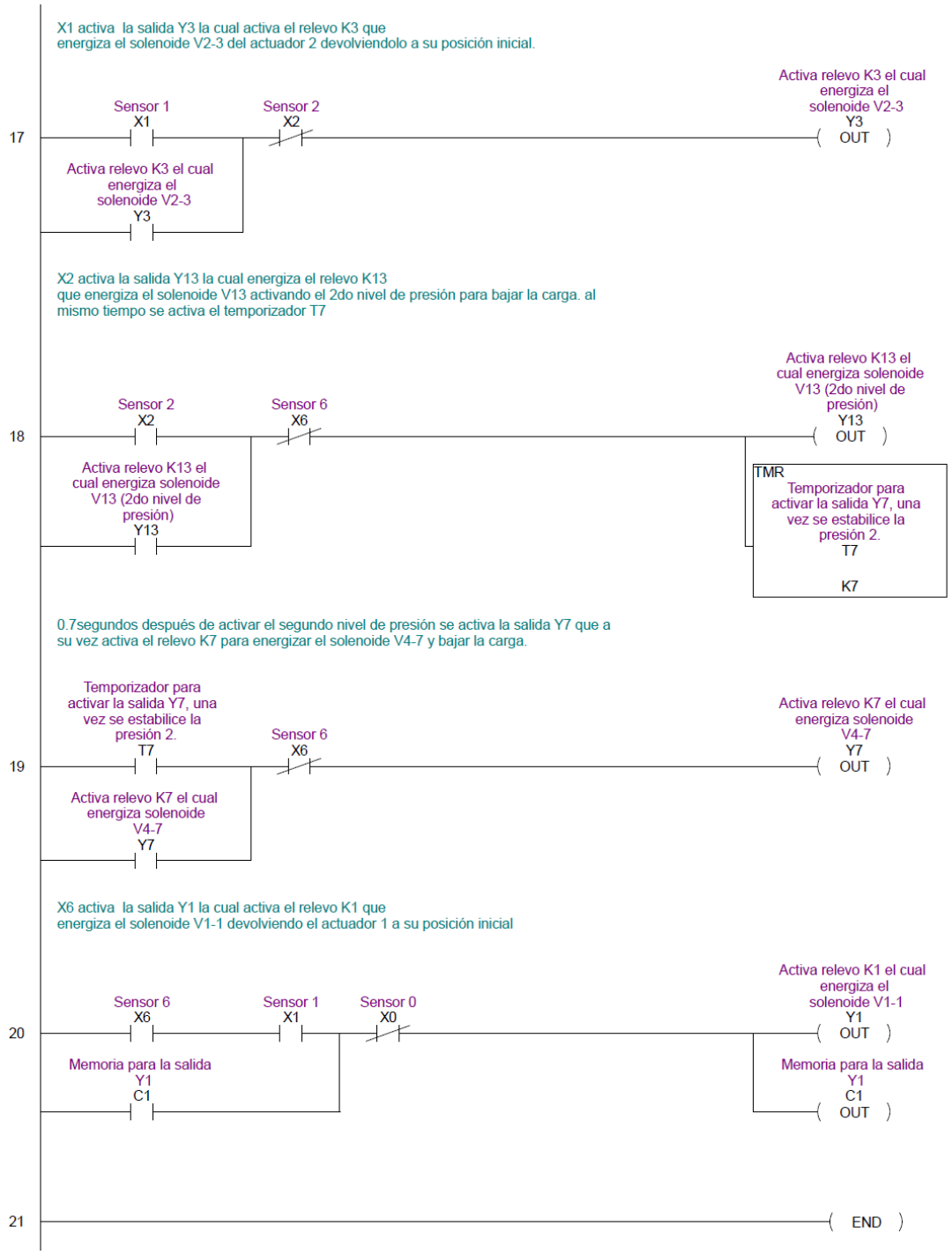
Figura 150. Ejemplo de programación del PLC de la cuarta práctica.



Fuente: Autores.


El contador CT22 desactiva la secuencia luego de una ejecución, los contactos C12 y X12 lo resetean,
CNT
Contador que permite controlar el número de veces que se ejecuta la secuencia.
CT2
K1
5
Sensor 2
X2
S12
X12
Resetea el contador una vez termianda la etapa de comprobación.
C12
SG
Activa la etapa S0
S0
Si S0 no detecta el actuador 1, lo debuelve a la posición inicial.
7
Sensor 0
X0
Activa la etapa de secuencias
S1
Activa relevo K1 el cual energiza el solenoide V1-1
Y1
OUT
La salida C12 es utilizada para resetear los contadores CT12 y CT2, ya que estos se pueden activar en etapa de comprobación, es decir vuelve los contadores a cero nuevamente para que se pueda ejecutar la secuencia.
8
Sensor 0
X0
Activa la etapa de secuencias
S1
Resetea el contador una vez termianda la etapa de comprobación.
C12
OUT
Sensor 2
X2
Si S2 no detecta el actuador 2, lo debuelve a la posición inicial.
9
Sensor 2
X2
Activa la etapa de secuencias
S1
Activa relevo K3 el cual energiza el solenoide V2-3
Y3
OUT
Si S6 no detecta el actuador 4, se activa el 2do nivel de presión por medio de Y13, y simultamenamente se activa el temporizador T70.
10
Sensor 6
X6
Sensor 0
X0
Sensor 2
X2
Activa la etapa de secuencias
S1
Activa relevo K13 el cual energiza solenoide V13 (2do nivel de presión)
Y13
OUT
TMR
Activa la salida Y7 luego de 0.7 seg
T70
K7


T70 activa la salida Y7, para bajar la carga, y se desactiva cuando el sensor 6 detecta la carga.
Activa la salida Y7 luego de 0.7 seg
T70
Sensor 6
X6
Activa relevo K7 el cual energiza solenoide V4-7
Y7
OUT
11
Una vez los actuadores están bien posicionados, se activa la etapa S1
Sensor 6
X6
Sensor 2
X2
Sensor 0
X0
Activa la etapa de secuencias
S1
JMP
12
Activa la etapa de secuencias
S1
S1 inicia la etapa de secuencia
SG
Activa la etapa de secuencias
S1
X2 activa la salida Y2 la cual activa el relevo K2 que energiza el solenoide V2-2 del actuador 2 moviendolo hacia afuera. los contactos X6, X0 garantizan que la secuencia no se repita hasta que sus respectivos actuadores esten en su posición inicial.
Contador que permite controlar el número de veces que se ejecuta la secuencia.
CT2
Sensor 2
X2
Sensor 6
X6
Sensor 0
X0
Sensor 3
X3
Activa relevo K2 para sacar el actuador 2.
Y15
OUT
14
Memoria para la salida
Y15
C15
Memoria para la salida
Y15
C15
OUT
X3 activa la salida Y6 la cual energiza el relevo K6 que energiza el solenoide V4-6 del actuador 4 subiendo la carga. el contacto T7 mantiene la carga suspendida hasta el momento de activar la salida Y7.
Temporizador para activar la salida Y7, una vez se estabilice la presión 2.
T7
Sensor 3
X3
Activa relevo K6 el cual energiza el solenoide V4-6
Y6
OUT
15
Activa relevo K6 el cual energiza el solenoide V4-6
Y6
X7 activa la salida Y0 la cual energiza el relevo K0 que energiza el solenoide V1-0 del actuador 1 moviendolo hacia afuera, Y0 se desactiva cuando el sensor 1 detecta el actuador.
Sensor 7
X7
Sensor 1
X1
Activa relevo K0 el cual energiza el solenoide V1-0
Y0
OUT
16
Activa relevo K0 el cual energiza el solenoide V1-0
Y0


X1 activa la salida Y3 la cual activa el relevo K3 que
energiza el solenoide V2-3 del actuador 2 devolviendolo a su posición inicial.
17
Sensor 1
X1
Sensor 2
X2
Activa relevo K3 el cual
energiza el
solenoide V2-3
Y3
OUT
Activa relevo K3 el cual
energiza el
solenoide V2-3
Y3
X2 activa la salida Y13 la cual energiza el relevo K13
que energiza el solenoide V13 activando el 2do nivel de presión para bajar la carga. al
mismo tiempo se activa el temporizador T7
18
Sensor 2
X2
Sensor 6
X6
Activa relevo K13 el
cual energiza solenoide
V13 (2do nivel de
presión)
Y13
OUT
Activa relevo K13 el
cual energiza solenoide
V13 (2do nivel de
presión)
Y13
TMR
Temporizador para
activar la salida Y7, una
vez se estabilice la
presión 2.
T7
K7
0.7segundos después de activar el segundo nivel de presión se activa la salida Y7 que a
su vez activa el relevo K7 para energizar el solenoide V4-7 y bajar la carga.
19
Temporizador para
activar la salida Y7, una
vez se estabilice la
presión 2.
T7
Sensor 6
X6
Activa relevo K7 el cual
energiza solenoide
V4-7
Y7
OUT
Activa relevo K7 el cual
energiza solenoide
V4-7
Y7
X6 activa la salida Y1 la cual activa el relevo K1 que
energiza el solenoide V1-1 devolviendo el actuador 1 a su posición inicial
20
Sensor 6
X6
Sensor 1
X1
Sensor 0
X0
Activa relevo K1 el cual
energiza el
solenoide V1-1
Y1
OUT
Memoria para la salida
Y1
C1
Memoria para la salida
Y1
C1
OUT
21
END

B. Microcontrolador

- Antes de realizar el programa lea la guía de programación para el Microcontrolador.
- Utilice el software Arduino para elaborar el programa.

Ejemplo de programación.

```
/*  Ejemplo  cuarta practica
 Cilindro número 2 ( + ),  malacate ( + ),
 Cilindro número 1 ( + ),  Cilindro número 2 ( - ),
 malacate ( - ), cilindro 1 ( - )
 NOTA:  *digitar el número de ciclos(cic) o cero(0) para repetir ciclos
indefinidamente
 *digitar los tiempos de retraso (t1mas, t1men, t2mas, t2men, t4mas, t4men)
 */
int cic = 3;
int t1mas = 0;     //tiempo de permanencia cilindro 1 +(milisegundos)
int t1men = 0;     //tiempo de permanencia cilindro 1 -(milisegundos)
int t2mas = 5;     //tiempo de permanencia cilindro 2 +(milisegundos)
int t2men = 0;     //tiempo de permanencia cilindro 2 -(milisegundos)
int t4mas = 0;     //tiempo de permanencia malacate +(milisegundos)
int t4men = 0;     //tiempo de permanencia malacate -(milisegundos)

//variables
int var1=0;         //variables de lectura  de la señal S10 (encendido de motor)
int var2=0;
int var3=0;      //variables de lectura  de la señal S11 (apagar motor)
int var4=0;

int S10 = A6;     // pulsador encendido entrada analógica  6
int S11 = A7;     // pulsador apagado entrada analógica  7
int senal_ens=0;    // almacena  señal encendido
int senal_apag=1;  // almacena  señal apagado
int on = 0;           // estado del motor
int K10 = 35;     // relevo del encendido modo estrella pin 35
int K11 = 33;     // relevo del apagado modo triangulo pin 33
int K12 = 31;       // 1er nivel de presión pin 31
int K13 = 29;      // 2do nivel de presión
int K14 = 27;      // alarma
int cont1=0;   // contador de ciclos
int cont2=0;   // contador para actuadores en posición inicial
int cont3=0;   //contador para impresión de número de ciclos
int Tref;    // tiempo de referencia para la alarma

// Cilindro 1
int S0 = A8;     //sensor 0 cilindro 1 ... entrada analógica pin 8
int S1 = A9;     //sensor cilindro 1 ... entrada analógica pin 9
int K0 = 51;     //válvula cilindro 1 +
int K1 = 49;     //válvula cilindro 1 -
int lS0 = 0;
int lS1 = 0;

// cilindro 2
int S2 = A0;     //sensor 2 cilindro 2 ... entrada analógica pin 2
```

```
int S3 = A1;     //sensor 3 cilindro 2 ... entrada analógica pin 3
int K2 = 47;     //válvula cilindro 1 +
int K3 = 45;     //válvula cilindro 1 -
int lS2 = 0;
int lS3 = 0;

//malacate
int S6 = A4;     //sensor 2 cilindro 2 ... entrada analógica pin 2
int S7 = A5;     //sensor 3 cilindro 2 ... entrada analógica pin 3
int K6 = 39;     //válvula malacate +
int K7 = 37;     //válvula malacate +
int lS6 = 0;     //almacena lectura sensor 6
int lS7 = 0;     //almacena lectura sensor 7

void setup(){
  Serial.begin(9600);       // comunicación

  pinMode(K10, OUTPUT);   //define como salida estrella
  pinMode(K11, OUTPUT);   //define como salida triangulo

  pinMode(K12, OUTPUT);   //define como salida presión 1
  pinMode(K13, OUTPUT);   //define como salida presión 2

  pinMode(K14, OUTPUT);   //define como salida la alarma

  //cilindro 1
  pinMode(K0, OUTPUT);   //define como salida electroválvula cil 1 +
  pinMode(K1, OUTPUT);   //define como salida electroválvula cil 1 -

  // cilindro 2
  pinMode(K2, OUTPUT);   //define como salida electroválvula cil 2 +
  pinMode(K3, OUTPUT);   //define como salida electroválvula cil 2 -

  // cmalacate
  pinMode(K6, OUTPUT);
  pinMode(K7, OUTPUT);


}
void loop(){
  encender();
  practica4();
  apagar();
}
void encender(){
  if(on == 0){           // si el motor está apagado entra
    var1 = analogRead(S10);  // lee estado de señal y almacena en var1
    if (var1 >= 500){      // convierte señal en cero o uno
      var1=1;
    }
    else {
      var1=0;
    }
    delay(10);             //retraso de 10 milisegundos
    var2 = analogRead(S10);  // lee estado de señal y almacena en var2
    if (var2 >= 500){      // convierte señal en cero o uno
      var2=1;
    }
    else {
      var2=0;
    }
```

```
    if (var1 == var2){      // evita rebote de contacto
      if (var1 != senal_ens){        // selecciona cuando hay  cambio de la señal
        if (var1 == LOW){    // selecciona cuando se suelta el pulsador
          digitalWrite(K14, LOW);
          digitalWrite(K10, HIGH);      //prende el motor modo estrella
          Serial.println("motor prendido modo estrella");   //imprime "motor
prendido modo estrella"
          delay(1500);       // espera 1,5 segundos
          digitalWrite(K11, HIGH);     //   pasa a modo triangulo
          Serial.println("motor prendido modo triangulo");   //imprime "motor
prendido modo triangulo"
          on = 1;
          delay(1000);     // retraso de un segundo
          cont1 = 0;
          cont2 = 0;
          cont3 = 0;
        }
        senal_ens = var1;
      }
    }
  }
}
void practica4(){
  if (cont2 < 1 && on == 1){
    digitalWrite(K12, HIGH);
    Serial.println("nivel presión 1");
    delay(700);
    posicioni1();
    posicioni2();
    posicioni4();
    ciclos();
    digitalWrite(K12, LOW);
    if (on == 1){
      Serial.println("fin de la tarea");
      digitalWrite(K12, LOW);
      Serial.println("venteo");
    }
    cont2++;
  }
}
void posicioni1(){  // función para posicionar cilindro 1
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 1");
  }
}
void posicioni2(){  // función para posicionar cilindro 2
  lS2 = analogRead(S2);
  Tref = millis();
  while (lS2 <= 500 && on == 1){
    digitalWrite(K3, HIGH);
    lS2 = analogRead(S2);
    alarma(10000, 2);
```

```
    apagar();
  }
  digitalWrite(K3, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 2");
  }
}
void posicioni4(){  // función para posicionar cilindro 4
  lS6 = analogRead(S6);
  if (lS6 <= 500 && on == 1){
    digitalWrite(K13, HIGH);
    Serial.println("nivel de presión dos");
    delay(700);
    Tref = millis();
    while (lS6 <= 500 && on == 1){
      digitalWrite(K7, HIGH);
      lS6 = analogRead(S6);
      alarma(10000, 4);
      apagar();
    }
    digitalWrite(K7, LOW);
    digitalWrite(K13, LOW);
    if(on == 1){
      Serial.println("nivel de presión uno");
    }
  }
  if(on == 1){
    Serial.println("posición correcta malacate");
  }
}
void ciclos(){   //función para el numero de ciclos
  while(cont1 < cic && on ==1){   // siempre que el contador de ciclos sea menor
que la cantidad de ciclos requeridos hace...
    cont3++;     // suma una unidad a el contador de ciclos para la impresión
    Serial.print("ciclo número ");
    Serial.println(cont3);        //   imprime en número de ciclo
    act_2_mas();
    act_4_mas();
    act_1_mas();
    act_2_men();
    act_4_men();
    act_1_men();
    if(cic != 0){ // si el número de ciclos es cero el ciclo se repetirá hasta
que se apague el motor
      cont1++;   //suma una unidad al número de ciclos
    }
  }
}
void act_1_mas(){   // función para cilindro 1( + )
  lS1 = analogRead(S1);
  Tref = millis();
  while (lS1 <= 500 && on == 1){
    digitalWrite(K0, HIGH);
    lS1 = analogRead(S1);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K0, LOW);
  fdelay(t1mas);
}
void act_1_men(){   // función para cilindro 1( - )
```

```
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  fdelay(t1men);
}
void act_2_mas(){   // función para cilindro 2( + )
  lS3 = analogRead(S3);
  Tref = millis();
  while (lS3 <= 500 && on == 1){
    digitalWrite(K2, HIGH);
    lS3 = analogRead(S3);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K2, LOW);
  fdelay(t2mas);
}

void act_2_men(){   // función para cilindro 2( - )
  lS2 = analogRead(S2);
  Tref = millis();
  while (lS2 <= 500 && on == 1){
    digitalWrite(K3, HIGH);
    lS2 = analogRead(S2);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K3, LOW);
  fdelay(t2men);
}
void act_4_mas(){   // función para malacate( + )
  lS7 = analogRead(S7);
  Tref = millis();
  while (lS7 <= 500 && on == 1){
    digitalWrite(K6, HIGH);
    lS7 = analogRead(S7);
    alarma(10000, 4);
    apagar();
  }
  fdelay(t4mas);
}

void act_4_men(){   // función para malacate( - )
  digitalWrite(K6, LOW);
  digitalWrite(K13, HIGH);
  if(on == 1){
    Serial.println("nivel de presión dos");
    delay(500);
  }
  lS6 = analogRead(S6);
  Tref = millis();
  while (lS6 <= 500 && on == 1){
    digitalWrite(K7, HIGH);
    lS6 = analogRead(S6);
    apagar();
```

```
    alarma(10000, 4);
  }
  digitalWrite(K7, LOW);
  digitalWrite(K13, LOW);
  if(on == 1){
    Serial.println("nivel de presión uno");
  }
  fdelay(t4men);
}
void apagar(){
  var3 = analogRead(S11);  // lee estado de señal y almacena en var3
  if (var3 >= 500){      // convierte señal en cero o uno
    var3=1;
  }
  else {
    var3=0;
  }
  delay(10);             //retraso de 10 milisegundos
  var4 = analogRead(S11);  // lee estado de señal y almacena en var4
  if (var4 >= 500){      // convierte señal en cero o uno
    var4=1;
  }
  else {
    var4=0;
  }
  if (var3 == var4){      // evita rebote de contacto
    if (var3 != senal_apag){        // selecciona cuando hay  cambio de la señal
      if (var3 == HIGH){
        digitalWrite(K14, LOW);
        if (on == 1){    // selecciona cuando se suelta el pulsador
          digitalWrite(K12, LOW);    // ventear
          digitalWrite(K10, LOW);    // apaga motor
          digitalWrite(K11, LOW);
          Serial.println("motor apagado");   //imprime "motor apagado"
          on = 0;
        }
      }
      senal_apag = var3;
    }
  }
}
void fdelay(int s){
  int a = 0;
  while (a <= s*1000 && on == 1){
    a = a + 100;
    delay (100);
    apagar();
  }
}

void alarma(int Talarma, int act){
  int T = -(Tref - millis());
  if (abs(T) >= Talarma && on == 1){
    digitalWrite(K14, HIGH);
    digitalWrite(K12, LOW);
    digitalWrite(K13, LOW);
    delay(1000);
    digitalWrite(K10, LOW);
    digitalWrite(K11, LOW);
    on = 0;
    Serial.print("ALERTA: hay problemas en el actuador ");
```

```
    Serial.println(act);
    Serial.println("Se recomienda revisar las electroválvulas y los sensores de
dicho actuador");
  }
}
```

8.8.8 Evaluación para los estudiantes.

1. Elabore un programa utilizando el lenguaje Ladder para el PLC y en C++ para el Microcontrolador que permita realizar la secuencia indicada en el ítem de programación.
2. ¿Cuántas fases tiene el motor eléctrico?
3. ¿Cuántos comunes tienen los puertos de entrada del PLC?
4. ¿Qué tipo de señales se están manejando analógicas o digitales?

8.9 PRÁCTICA 5: ARRANQUE DEL MOTOR Y ACCIONAMIENTO DE LOS ACTUADORES 1, 2, 3 y 4

8.9.1 Objetivos

- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo utilizando el lenguaje Ladder aplicado al PLC.
- Programar el encendido de un motor trifásico por medio del sistema Estrella-triangulo utilizando el lenguaje C++ aplicado al Microcontrolador.
- Programar el accionamiento de los actuadores 1, 2, 3 y 4 utilizando el lenguaje Ladder aplicado al PLC.
- Programar el accionamiento de los actuadores 1, 2, 3 y 4 utilizando el lenguaje C++ aplicado al Microcontrolador.

8.9.2 Fundamentación previa

- Revisar manual teórico.
- Identificar las principales características de los equipos utilizados en la elaboración de la práctica.

8.9.3 Procedimientos de seguridad. Antes de la elaboración de las prácticas se deben tener previo conocimiento de los procedimientos de seguridad que se deben seguir, estos se exponen en el inicio de este manual.

8.9.4 Ubicación e identificación de los componentes. Identifique cada uno de los componentes mencionados en la tabla 25. Para ver la distribución de los elementos ubicados en el panel ver la figura 151.

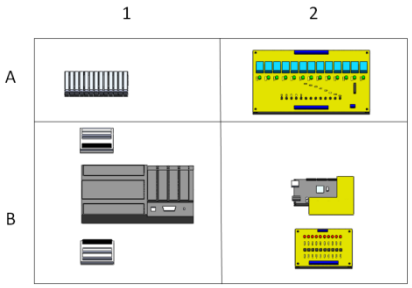
Tabla 25. Componentes requeridos para la elaboración de la práctica 5.

ELEMENTO	NOMENCLATURA	UBICACIÓN
Parada de emergencia	-S20	Tablero de mando
Selector de energía principal	-S21	Tablero de mando
Indicador de energía principal	-H1	Tablero de mando
Selector de equipo (PLC o Microcontrolador)	-S22	Tablero de mando
Indicador del PLC	-H5	Tablero de mando
Indicador de Microncontrolador	-H6	Tablero de mando
Pulsador inicio del motor	-S10	Tablero de mando
Pulsador apagado del motor	-S11	Tablero de mando
Indicador motor encendido	-H2	Tablero de mando
Indicador de válvula de presión 1	-H3	Tablero de mando
PLC		B1
Conectores de entradas al PLC	C1	B1
Conectores de salidas del PLC	C2	B1
Microcontrolador		B2
Tarjeta de interface de entadas		B1
Tarjeta de potencia		A2
Bornes portafusibles	-F	A1
Microcontrolador		B2
Contactores arranque del motor	-KM1, -KM2, -KM3	
Relevos para arranque del motor	-K10, -K11 -KA10, -KA11	Tarjeta de potencia Tablero de potencia
Relevo para activación de 1^{er} nivel de presión	-K12	Tarjeta de potencia
Relevo para activación de 2^{er} nivel de presión	-K13	Tarjeta de potencia
Relevo para activación solenoide V1-0 de la válvula 1	-k0	Tarjeta de potencia
Relevo para activación solenoide V1-1 de la válvula 1	-k1	Tarjeta de potencia
Relevo para activación solenoide V2-2 de la válvula 2	-k2	Tarjeta de potencia
Relevo para activación solenoide V2-3 de la válvula 2	-k3	Tarjeta de potencia
Relevo para activación solenoide V3-4 de la válvula 3	-k4	Tarjeta de potencia
Relevo para activación solenoide V3-5 de la válvula 3	-k5	Tarjeta de potencia
Relevo para activación solenoide V4-6 de la válvula 4	-k6	Tarjeta de potencia
Relevo para activación solenoide V4-7 de la válvula 4	-k7	Tarjeta de potencia
Sensor 0	-S0	Inicio de carrera del actuador 1

Fuente: Autores.

Sensor 1	-S1	final de carrera del actuador 1
Sensor 2	-S2	Inicio de carrera del actuador 2
Sensor 3	-S3	final de carrera del actuador 2
Sensor 4	-S4	Inicio de carrera del actuador 3
Sensor 5	-S5	final de carrera del actuador 5
Sensor 6	-S6	Inicio de carrera de la carga
Sensor 7	-S7	final de carrera de la carga
Válvula primer nivel de presión	-S12	Unidad hidráulica
Válvula 1	V1	Unidad hidráulica
Solenoides válvula 1	V1-0 (sale actuador) V1-1 (entra actuador)	Unidad hidráulica
Válvula 2	V2	Unidad hidráulica
Solenoides válvula 2	V2-2 (sale actuador) V2-3 (entra actuador)	Unidad hidráulica
Válvula 3	V3	Unidad hidráulica
Solenoides válvula 3	V3-4 (sale actuador) V3-5 (entra actuador)	Unidad hidráulica
Válvula 4	V3	Unidad hidráulica
Solenoides válvula 4	V4-6 (sube la carga) V4-7 (baja la carga)	Unidad hidráulica
Actuador 1		Unidad hidráulica
Actuador 2		Unidad hidráulica
Actuador 3		Unidad hidráulica
Actuador 4		Unidad hidráulica
Válvula de 1er nivel de presión	V12	Unidad hidráulica
Válvula de 2do nivel de presión	V13	Unidad hidráulica

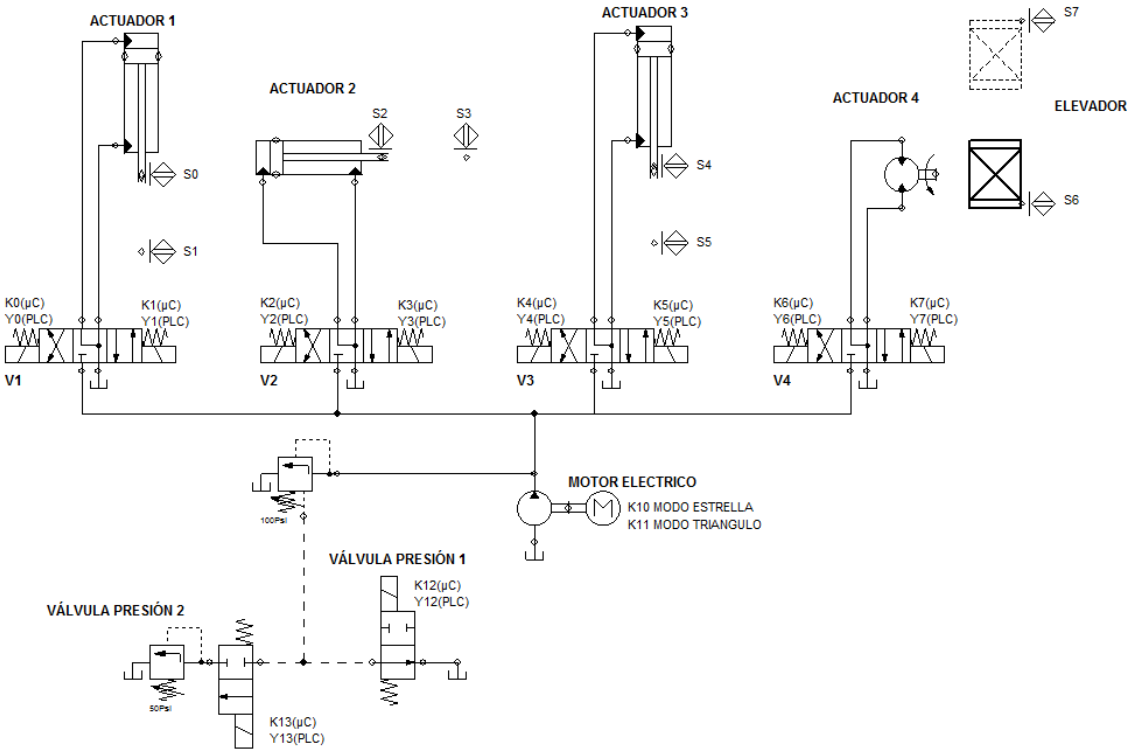
Figura 151. Distribución del los elementos en el panel.



Fuente: Autores.

8.9.5 Reconocimiento del circuito hidráulico. El circuito hidráulico a controlar consta de un motor eléctrico, dos electroválvulas de presión, tres actuadores tipo cilindro de doble efecto controlados cada uno por una electroválvula y un motor hidráulico que acciona un malacate. La figura 152 ilustra el circuito hidráulico.

Figura 152. Circuito hidráulico de la cuarta práctica.



Autores: Autores.

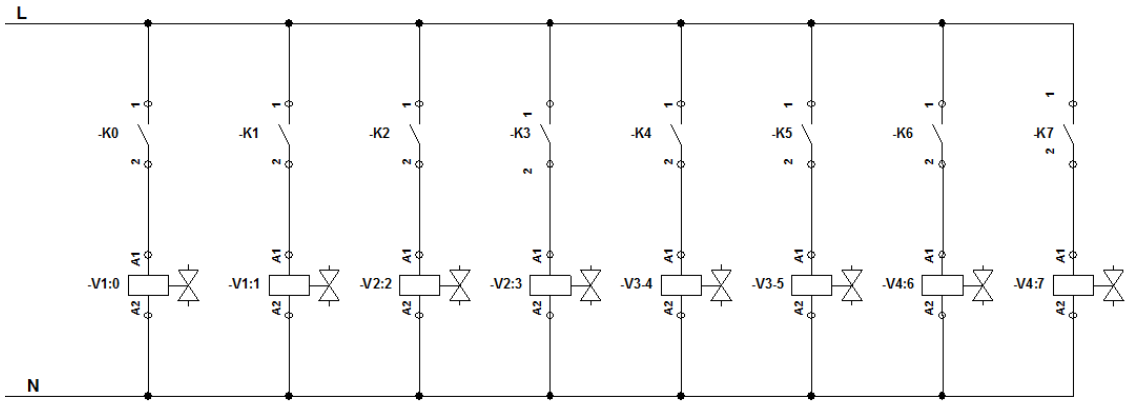
8.9.6 Reconocimiento de los circuitos eléctricos. Analice y discuta cada uno de los circuitos eléctricos requeridos para la elaboración de la práctica.

Ver figura 119, 120, 121, 122

.

En la figura 153, se puede observar el esquema eléctrico para el accionamiento de las electroválvulas 1, 2, 3 y 4 requeridas para la elaboración de la práctica.

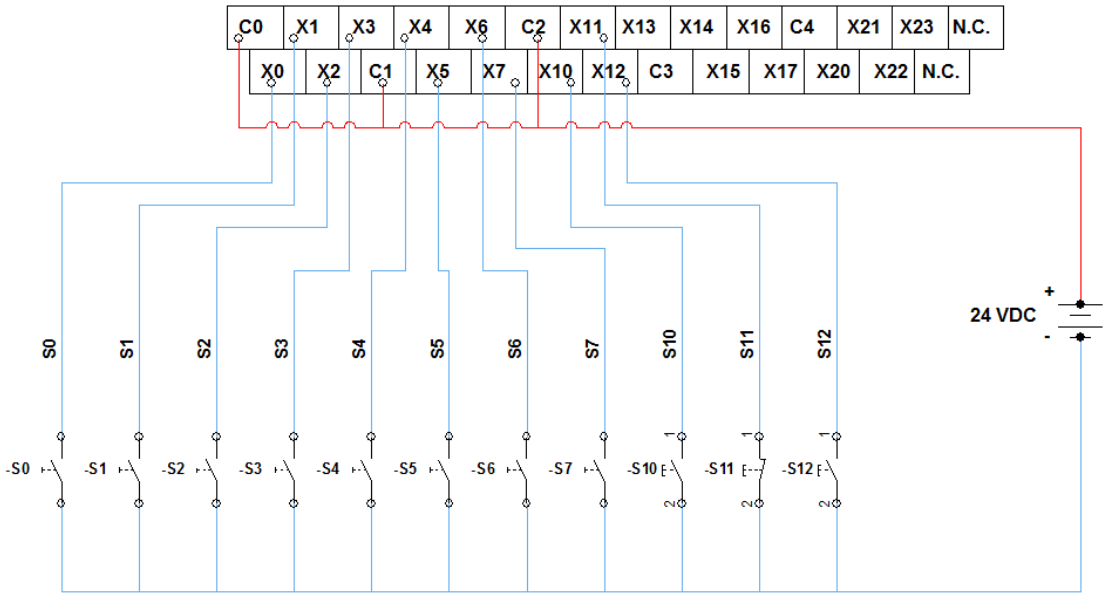
Figura 153. Esquema eléctrico del accionamiento de las electroválvulas 1, 2, 3 y 4.



Fuente: Autores.

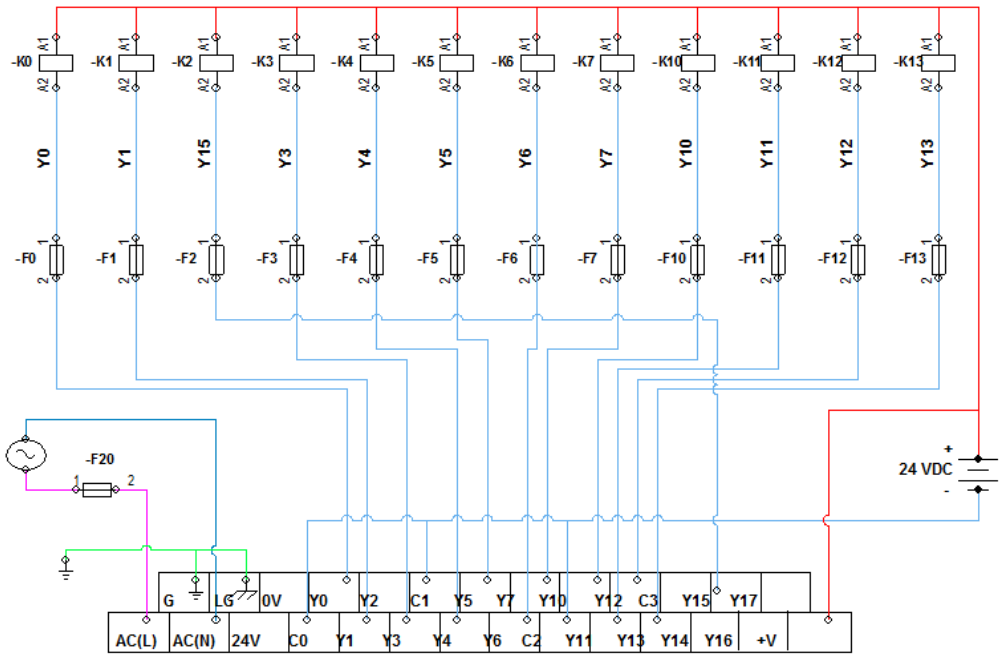
En las figuras 154 y 155 se observan las conexiones de entradas y salidas respectivamente del PLC que se requieren para la elaboración de la práctica 4.

Figura 154. Esquema eléctrico de las conexiones de entrada del PLC requeridas para la práctica cinco.



Fuente: Autores.

Figura 155. Esquema eléctrico de las conexiones de salida del PLC requeridas para la práctica cinco.



Fuente: Autores.

La tabla 26 muestra los puertos de salidas del Microncontrolador y los relevos que activan en la tarjeta de potencia.

Tabla 26. Puertos de salidas de Microcontrolador y los relevos que activan en la tarjeta de potencia (Práctica cinco).

PUERTOS DIGITALES DE LA TARJETA ARDUINO	SEÑALES DE SALIDA-TARJETA DE POTENCIA
29	K13
31	K12
33	K11
35	K10
37	K7
39	K6
41	K5
43	K4

Fuente: Autores

45	**K3**
47	**K2**
49	**K1**
51	**K0**

La tabla 27 muestra los puertos de entrada del Microncontrolador y las señales de los sensores que llegan a estos puertos.

Tabla 27. Puertos de entrada del Microncontrolador y señales de los sensores que llegan a los mismos (Práctica cinco).

PUERTOS ANALÓGICOS DE LATARJETA ARDUINO	SEÑALES DE ENTRADA-TARJETA INTERFAZ
A0	**S2**
A1	**S3**
A2	**S4**
A3	**S5**
A4	**S6**
A5	**S7**
A6	**S10**
A7	**S11**
A8	**S0**
A9	**S1**

Fuente: Autores.

8.9.7 Programación

Programe la siguiente secuencia para el PLC y el Microncontrolador.

a) Al presionar S10 se enciende motor por medio del relevo K10.

b) 1.5 segundos después encender el relevo K11 para pasar de estrella a

c) triangulo.

d) 1 segundo después active el 1er nivel de presión activando el relevo K12.

e) Compruebe que los actuadores 1, 2 y 4 se encuentren en posición correcta.

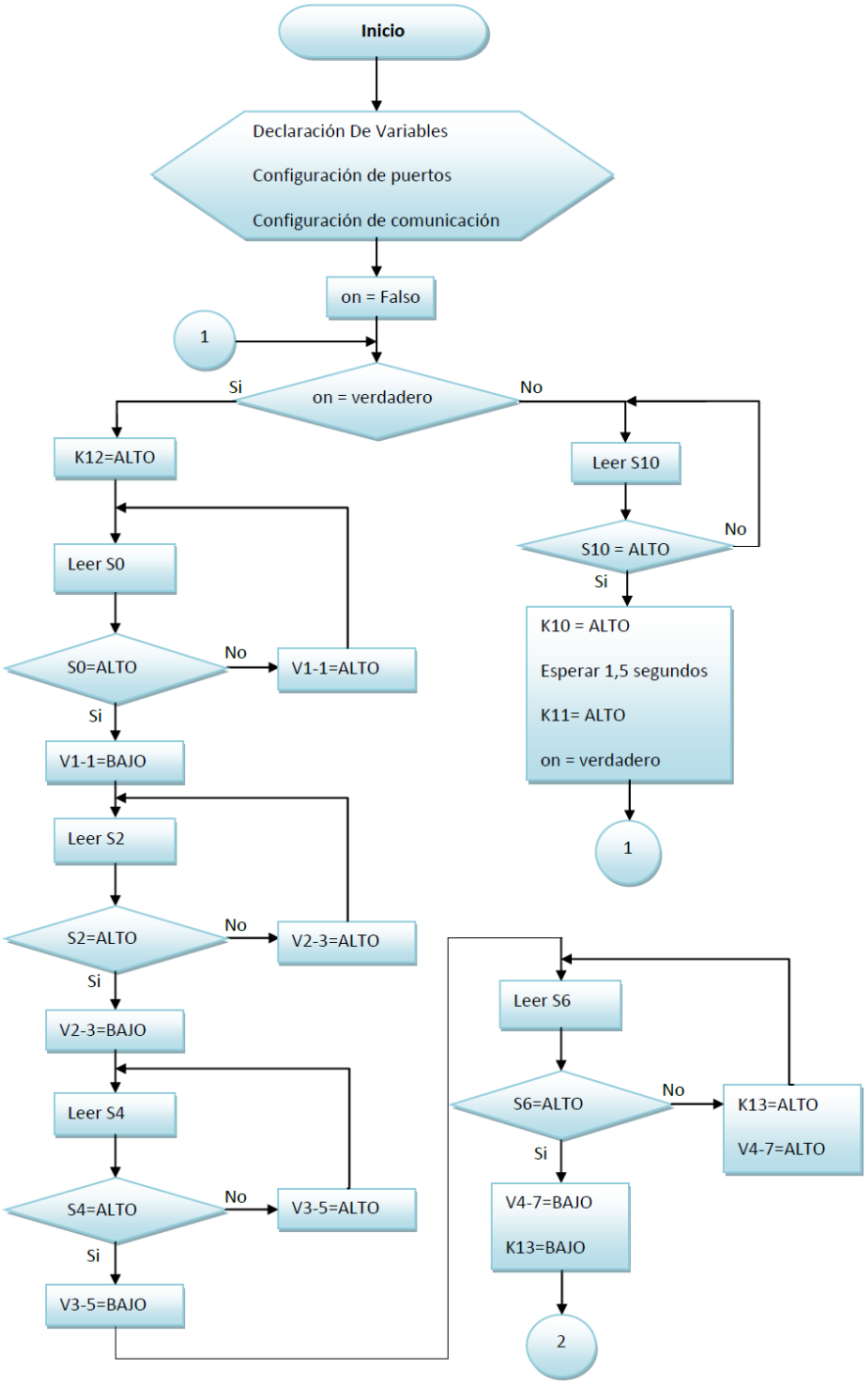
f) Realice la siguiente secuencia:

1. Active solenoide V4-6 para subir la carga.

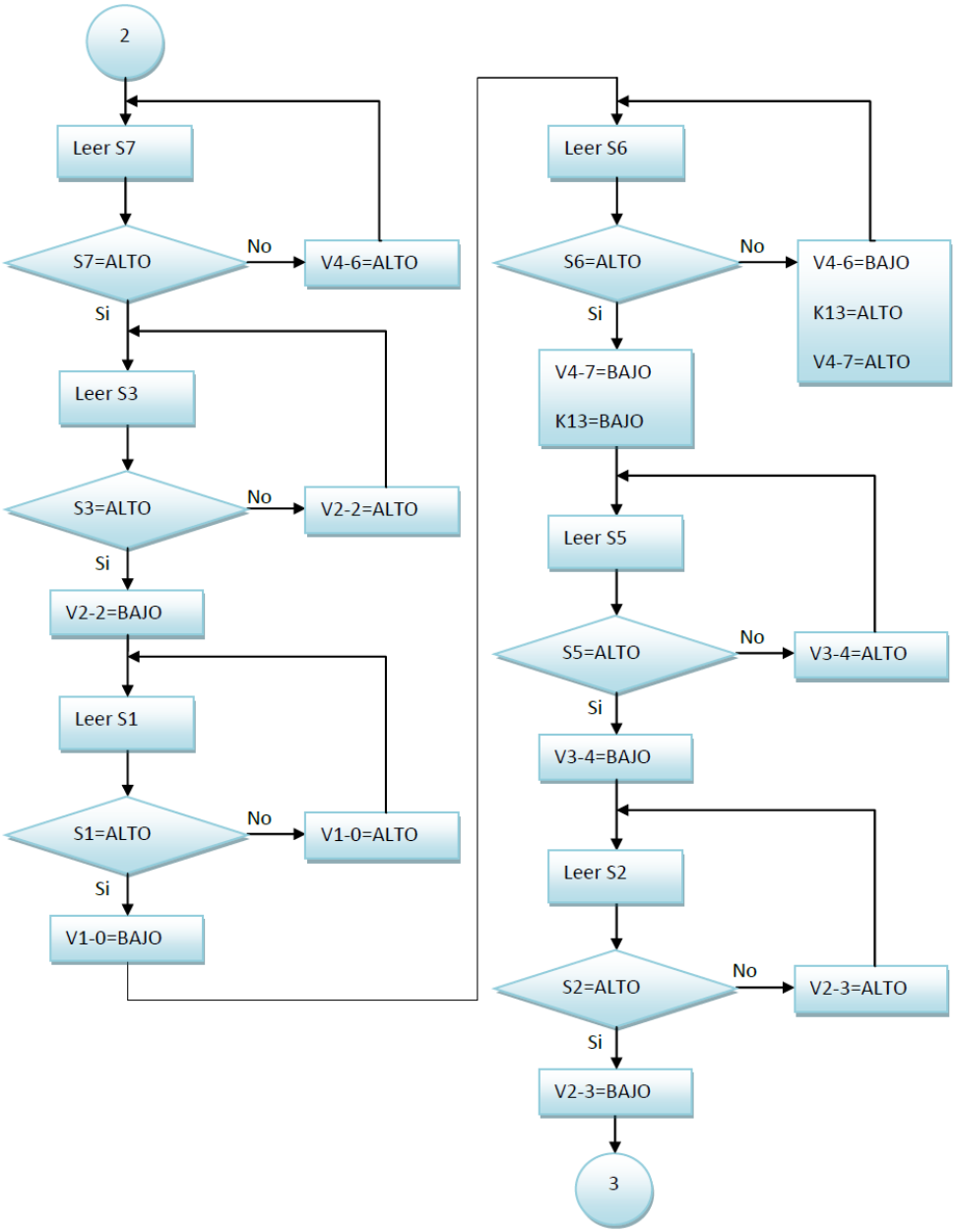
2. Active solenoide V2-2 para sacar el actuador 2.
3. Active solenoide V0-0 para sacar el actuador 1.
4. Active solenoide V13 para activar 2do nivel de presión.
5. 0.7 segundos después active solenoide V4-7 para bajar la carga.
6. Active solenoide V3-4 para sacar el actuador 3.
7. Active solenoide V2-3 para devolver a su posición inicial el actuador 2.
8. Active solenoide V0-1 para devolver a su posición inicial el actuador 1.
9. Active solenoide V3-5 para devolver a su posición inicial el actuador 3.
10. Desactive 1er nivel de presión.
11. Apague el motor.

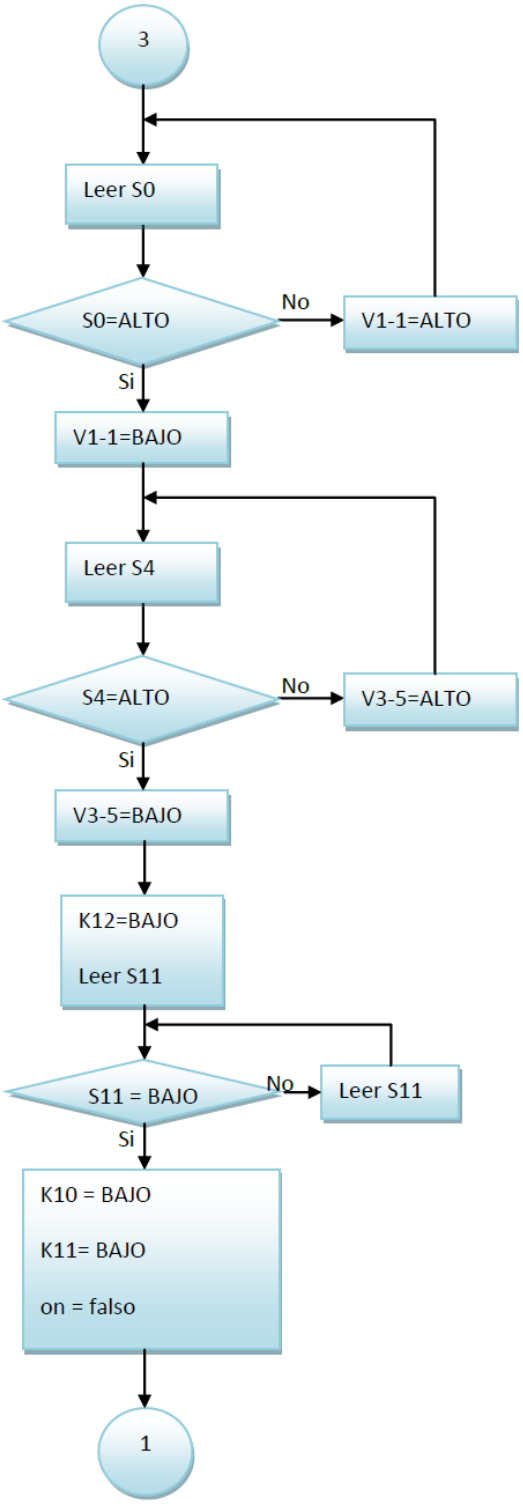
En la figura 156 se muestra un ejemplo del diagrama de flujo del programa para la quinta práctica.

Figura 156. Diagrama de flujo del programa de quinta práctica.



Fuente: Autores.


2
Leer S7
S7=ALTO
No
V4-6=ALTO
Si
Leer S3
S3=ALTO
No
V2-2=ALTO
Si
V2-2=BAJO
Leer S1
S1=ALTO
No
V1-0=ALTO
Si
V1-0=BAJO
Leer S6
S6=ALTO
No
V4-6=BAJO
K13=ALTO
V4-7=ALTO
Si
V4-7=BAJO
K13=BAJO
Leer S5
S5=ALTO
No
V3-4=ALTO
Si
V3-4=BAJO
Leer S2
S2=ALTO
No
V2-3=ALTO
Si
V2-3=BAJO
3


3
Leer S0
S0=ALTO
No
V1-1=ALTO
Si
V1-1=BAJO
Leer S4
S4=ALTO
No
V3-5=ALTO
Si
V3-5=BAJO
K12=BAJO
Leer S11
S11 = BAJO
No
Leer S11
Si
K10 = BAJO
K11= BAJO
on = falso
1

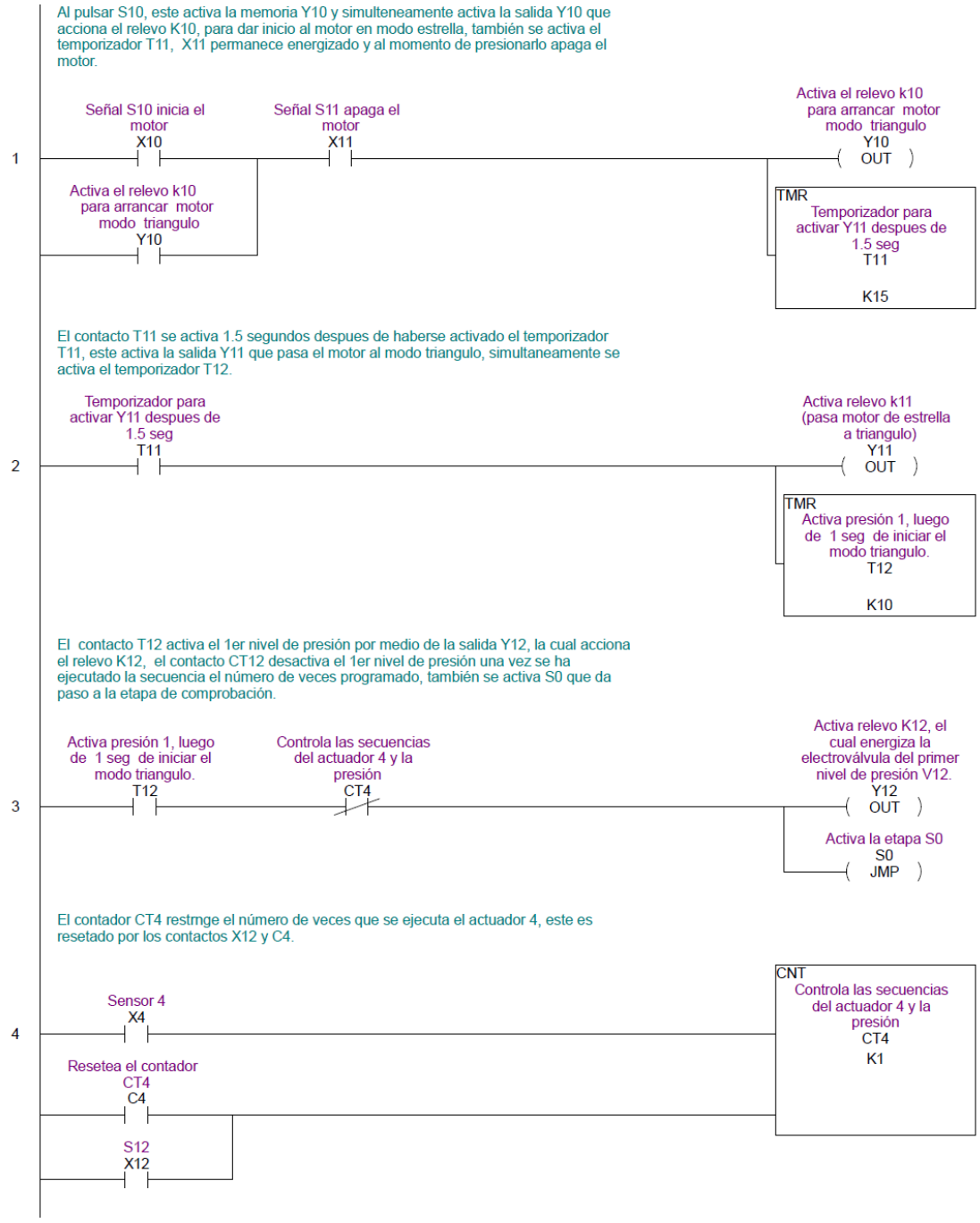
A. PLC

- Antes de realizar el programa lea la guía de programación para el PLC.
- Utilice el software DirectSOFT 5 para elaborar el programa.

Ejemplo de programación.

En la figura 157 se muestra un ejemplo de programación para el PLC de la quinta práctica.

Figura 157. Ejemplo de programación del PLC de la quinta práctica.



Fuente: Autores.

El contador CT6 restrnge el número de veces que se ejecuta el actuador 6, este es resetado por los contactos X12 y C6.

5

Sensor 6
X6

S12
X12

CNT
Contador que permite controlar el número de veces que se ejecuta la secuencia.
CT6
K2

El contador CT41impide que el actuador 4 realice lasecuencia antes de que la relaice el actuador 6, este es activado unicamente por el sensor 6, y reseteado por el sensor 7.

6

Sensor 6
X6

Sensor 7
X7

CNT
Se activa cuando S7 es detectado
CT41
K1

S0 da inicio a la etapa de comprobación.

SG
Activa la etapa S0
S0

Si S0 no detecta el actuador 1, lo debuelve a la posición inicial.

8

Sensor 0
X0

Activa la etapa S1
S1

Activa relevo K1, el cual energiza el solenoide V1-1
Y1
OUT

Si S2 no detecta el actuador 2, lo debuelve a la posición inicial.

9

Sensor 2
X2

Activa la etapa S1
S1

Activa relevo K3, el cual energiza solenoide V2-3
Y3
OUT

Si S4 no detecta el actuador 3, lo debuelve a la posición inicial. para que ello suceda los actuadores 1,2 y la carga deben estar en posición correcta.

10

Sensor 4
X4

Sensor 0
X0

Sensor 2
X2

Sensor 6
X6

Activa la etapa S1
S1

Activa relevo K5, el cual energiza la vávula V4-5
Y5
OUT

La salida C4 es utilizada para resetear el contador CT4, ya que esto se puede activar en etapa de comprobación, es decir vuelve el contador a cero nuevamente para que se pueda ejecutar la secuencia.

11

Sensor 4
X4

Activa la etapa S1
S1

Resetea el contador CT4
C4
OUT

Una vez todos los actuadores se encuentren en posición correcta, se activa S1 para dar paso a la etapa de secuencia.

12

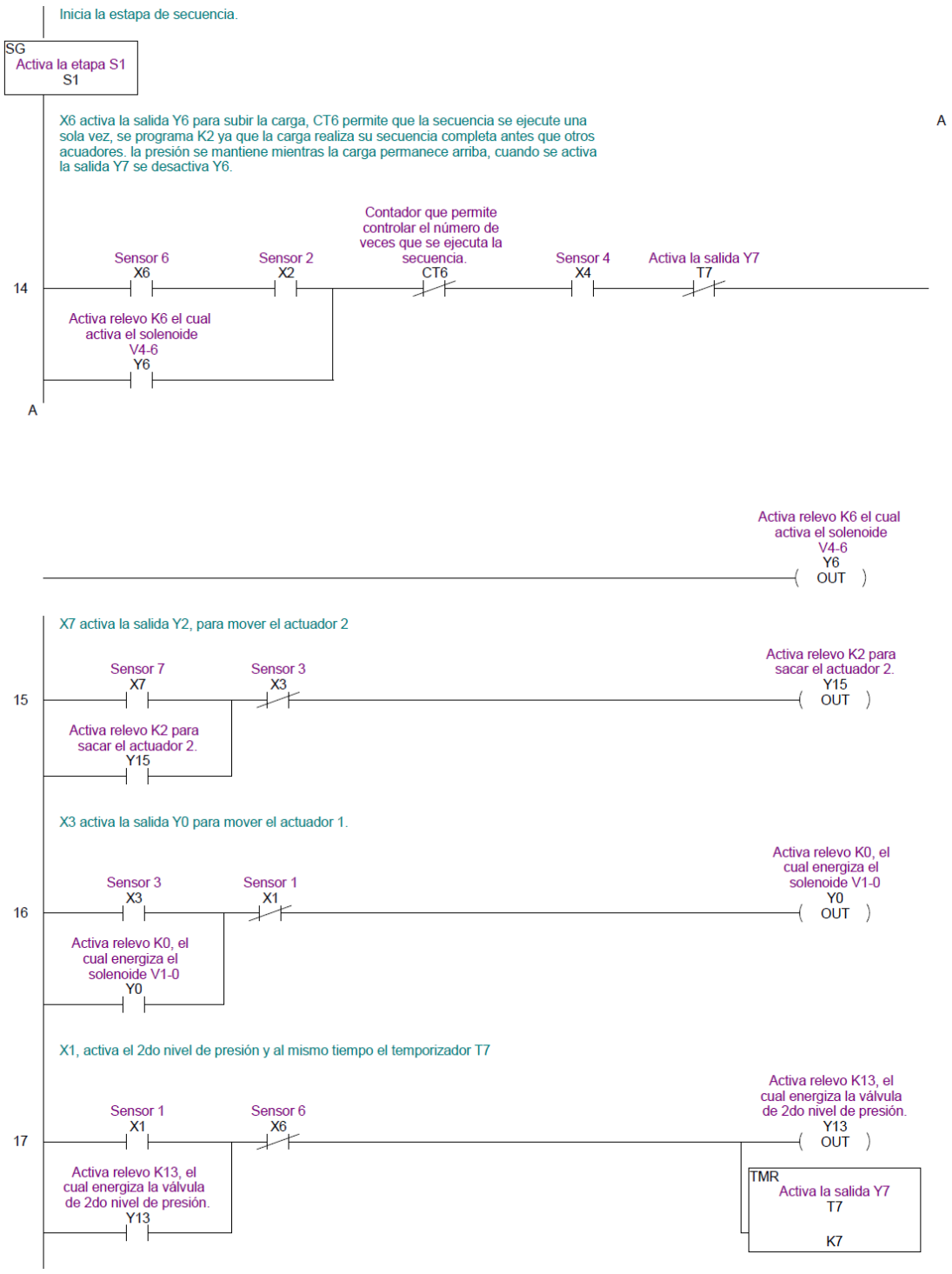
Sensor 4
X4

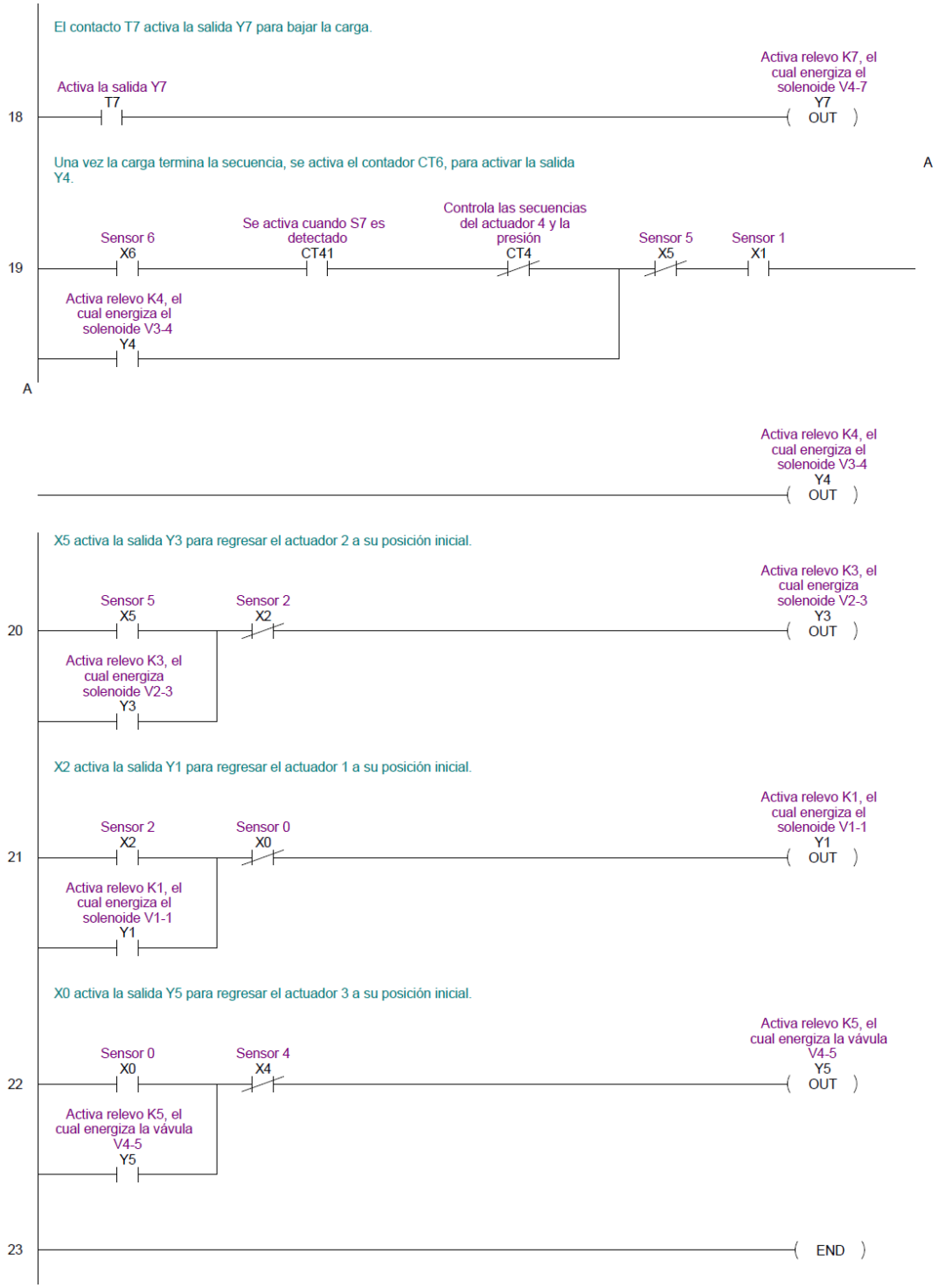
Sensor 2
X2

Sensor 0
X0

Activa la etapa S1
S1
JMP

Activa la etapa S1
S1


Inicia la estapa de secuencia.
SG
Activa la etapa S1
S1
X6 activa la salida Y6 para subir la carga, CT6 permite que la secuencia se ejecute una sola vez, se programa K2 ya que la carga realiza su secuencia completa antes que otros acuadores. la presión se mantiene mientras la carga permanece arriba, cuando se activa la salida Y7 se desactiva Y6.
A
Contador que permite controlar el número de veces que se ejecuta la secuencia.
Sensor 6
X6
Sensor 2
X2
CT6
Sensor 4
X4
Activa la salida Y7
T7
14
Activa relevo K6 el cual activa el solenoide V4-6
Y6
A
Activa relevo K6 el cual activa el solenoide V4-6
Y6
OUT
X7 activa la salida Y2, para mover el actuador 2
Sensor 7
X7
Sensor 3
X3
Activa relevo K2 para sacar el actuador 2.
Y15
OUT
15
Activa relevo K2 para sacar el actuador 2.
Y15
X3 activa la salida Y0 para mover el actuador 1.
Activa relevo K0, el cual energiza el solenoide V1-0
Y0
OUT
Sensor 3
X3
Sensor 1
X1
16
Activa relevo K0, el cual energiza el solenoide V1-0
Y0
X1, activa el 2do nivel de presión y al mismo tiempo el temporizador T7
Activa relevo K13, el cual energiza la válvula de 2do nivel de presión.
Y13
OUT
Sensor 1
X1
Sensor 6
X6
17
Activa relevo K13, el cual energiza la válvula de 2do nivel de presión.
Y13
TMR
Activa la salida Y7
T7
K7


El contacto T7 activa la salida Y7 para bajar la carga.
Activa la salida Y7
T7
Activa relevo K7, el cual energiza el solenoide V4-7
Y7
OUT
18
Una vez la carga termina la secuencia, se activa el contador CT6, para activar la salida Y4.
A
Sensor 6
X6
Se activa cuando S7 es detectado
CT41
Controla las secuencias del actuador 4 y la presión
CT4
Sensor 5
X5
Sensor 1
X1
19
Activa relevo K4, el cual energiza el solenoide V3-4
Y4
A
Activa relevo K4, el cual energiza el solenoide V3-4
Y4
OUT
X5 activa la salida Y3 para regresar el actuador 2 a su posición inicial.
Sensor 5
X5
Sensor 2
X2
Activa relevo K3, el cual energiza solenoide V2-3
Y3
OUT
20
Activa relevo K3, el cual energiza solenoide V2-3
Y3
X2 activa la salida Y1 para regresar el actuador 1 a su posición inicial.
Sensor 2
X2
Sensor 0
X0
Activa relevo K1, el cual energiza el solenoide V1-1
Y1
OUT
21
Activa relevo K1, el cual energiza el solenoide V1-1
Y1
X0 activa la salida Y5 para regresar el actuador 3 a su posición inicial.
Sensor 0
X0
Sensor 4
X4
Activa relevo K5, el cual energiza la vávula V4-5
Y5
OUT
22
Activa relevo K5, el cual energiza la vávula V4-5
Y5
23
END

B. Microcontrolador

- Antes de realizar el programa lea la guía de programación para el Microcontrolador.
- Utilice el software Arduino para elaborar el programa.

Ejemplo de programación

```
/*  Ejemplo  quinta practica
 malacate ( + ), Cilindro número 2 ( + ),
 Cilindro número 1 ( + ), malacate ( - ),
 Cilindro número 3 ( + ), Cilindro número 2 ( - ),
 cilindro 1 ( - ), Cilindro número 3 ( - )
 NOTA:  *digitar el número de ciclos(cic) o cero(0) para repetir ciclos
indefinidamente
 *digitar los tiempos de retraso (t1mas, t1men, t2mas, t2men, t4mas, t4men)
 */
int cic = 3;
int t1mas = 0;     //tiempo de permanencia cilindro 1 +(milisegundos)
int t1men = 0;     //tiempo de permanencia cilindro 1 -(milisegundos)
int t2mas = 0;     //tiempo de permanencia cilindro 2 +(milisegundos)
int t2men = 0;     //tiempo de permanencia cilindro 2 -(milisegundos)
int t3mas = 0;     //tiempo de permanencia cilindro 3 +(milisegundos)
int t3men = 0;     //tiempo de permanencia cilindro 3 -(milisegundos)
int t4mas = 0;     //tiempo de permanencia malacate +(milisegundos)
int t4men = 0;     //tiempo de permanencia malacate -(milisegundos)

//variables
int var1=0;         //variables de lectura  de la señal S10 (encendido de motor)
int var2=0;
int var3=0;      //variables de lectura  de la señal S11 (apagar motor)
int var4=0;

int S10 = A6;     // pulsador encendido entrada analógica  6
int S11 = A7;     // pulsador apagado entrada analógica  7
int senal_ens=0;    // almacena  señal encendido
int senal_apag=1;  // almacena  señal apagado
int on = 0;           // estado del motor
int K10 = 35;     // relevo del encendido modo estrella pin 35
int K11 = 33;     // relevo del apagado modo triangulo pin 33
int K12 = 31;       // 1er nivel de presión pin 31
int K13 = 29;      // 2do nivel de presiónK2
int K14 = 27;      // alarma
int cont1=0;   // contador de ciclos
int cont2=0;   // contador para actuadores en posición inicial
int cont3=0;   //contador para impresión de número de ciclos
int Tref;    // tiempo de referencia para la alarma

// Cilindro 1
int S0 = A8;     //sensor 0 cilindro 1 ... entrada analógica pin 8
int S1 = A9;     //sensor cilindro 1 ... entrada analógica pin 9
int K0 = 51;     //válvula cilindro 1 +
int K1 = 49;     //válvula cilindro 1 -
int lS0 = 0;     //almacena lectura sensor 0
int lS1 = 0;     //almacena lectura sensor 1
// cilindro 2
```

```
int S2 = A0;     //sensor 2 cilindro 2 ... entrada analógica pin 2
int S3 = A1;     //sensor 3 cilindro 2 ... entrada analógica pin 3
int K2 = 47;     //válvula cilindro 1 +
int K3 = 45;     //válvula cilindro 1 -
int lS2 = 0;     //almacena lectura sensor 0
int lS3 = 0;     //almacena lectura sensor 1

//cilindro 3
int S4 = A2;     //sensor 4 cilindro 3 ... entrada analógica pin 4
int S5 = A3;     //sensor 5 cilindro 3 ... entrada analógica pin 5
int K4 = 43;     //válvula cilindro 3 +
int K5 = 41;     //válvula cilindro 3 +
int lS4 = 0;     //almacena lectura sensor 4
int lS5 = 0;     //almacena lectura sensor 5

//malacate
int S6 = A4;     //sensor 2 cilindro 2 ... entrada analógica pin 2
int S7 = A5;     //sensor 3 cilindro 2 ... entrada analógica pin 3
int K6 = 39;     //válvula malacate +
int K7 = 37;     //válvula malacate +
int lS6 = 0;     //almacena lectura sensor 6
int lS7 = 0;     //almacena lectura sensor 7

void setup(){            // configuración de puertos y de comunicación
  Serial.begin(9600);       // comunicación

  pinMode(K10, OUTPUT);   //define como salida estrella
  pinMode(K11, OUTPUT);   //define como salida triangulo

  pinMode(K12, OUTPUT);   //define como salida presión 1
  pinMode(K13, OUTPUT);   //define como salida presión 2

  pinMode(K14, OUTPUT);   //define como salida la alarma

  //cilindro 1
  pinMode(K0, OUTPUT);   //define como salida electroválvula cil 1 +
  pinMode(K1, OUTPUT);   //define como salida electroválvula cil 1 -

  // cilindro 2
  pinMode(K2, OUTPUT);   //define como salida electroválvula cil 2 +
  pinMode(K3, OUTPUT);   //define como salida electroválvula cil 2 -

  // cilindro 3
  pinMode(K4, OUTPUT);   //define como salida electroválvula cil 3 +
  pinMode(K5, OUTPUT);   //define como salida electroválvula cil 3 -

  // malacate
  pinMode(K6, OUTPUT);   //define como salida electroválvula malacate +
  pinMode(K7, OUTPUT);    //define como salida electroválvula malacate +
}
void loop(){
  encender();
  practica();
  apagar();
}
void encender(){
  if(on == 0){           // si el motor está apagado entra
    var1 = analogRead(S10);  // lee estado de señal y almacena en var1
    if (var1 >= 500){      // convierte señal en cero o uno
      var1=1;
```

```
    }
    else {
      var1=0;
    }
    delay(10);             //retraso de 10 milisegundos
    var2 = analogRead(S10);  // lee estado de señal y almacena en var2
    if (var2 >= 500){      // convierte señal en cero o uno
      var2=1;
    }
    else {
      var2=0;
    }
    if (var1 == var2){      // evita rebote de contacto
      if (var1 != senal_ens){        // selecciona cuando hay  cambio de la señal
        if (var1 == LOW){    // selecciona cuando se suelta el pulsador
          digitalWrite(K14, LOW);
          digitalWrite(K10, HIGH);      //prende el motor modo estrella
          Serial.println("motor prendido modo estrella");   //imprime "motor
prendido modo estrella"
          delay(1500);       // espera 1,5 segundos
          digitalWrite(K11, HIGH);     //   pasa a modo triangulo
          Serial.println("motor prendido modo triangulo");   //imprime "motor
prendido modo triangulo"
          on = 1;
          delay(1000);     // retraso de un segundo
          cont1 = 0;
          cont2 = 0;
          cont3 = 0;
        }
        senal_ens = var1;
      }
    }
  }
}
void practica(){
  if (cont2 < 1 && on == 1){
    digitalWrite(K12, HIGH);
    Serial.println("nivel presión 1");
    delay(700);
    posicioni1();
    posicioni2();
    posicioni3();
    posicioni4();
    ciclos();
    if (on == 1){
      Serial.println("fin de la tarea");
     digitalWrite(K12, LOW;
      Serial.println("venteo");
    }
    cont2++;
  }
}
void posicioni1(){  // función para posicionar cilindro 1
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
```

```
  }
  digitalWrite(K1, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 1");
  }
}
void posicioni2(){  // función para posicionar cilindro 2
  lS2 = analogRead(S2);
  Tref = millis();
  while (lS2 <= 500 && on == 1){
    digitalWrite(K3, HIGH);
    lS2 = analogRead(S2);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K3, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 2");
  }
}
void posicioni3(){  // función para posicionar cilindro 3
  lS4 = analogRead(S4);
  Tref = millis();
  while (lS4 <= 500 && on == 1){
    digitalWrite(K5, HIGH);
    lS4 = analogRead(S4);
    alarma(50000, 3);
    apagar();
  }
  digitalWrite(K5, LOW);
  if(on == 1){
    Serial.println("posición correcta cilindro 3");
  }
}
void posicioni4(){  // función para posicionar cilindro 4
  lS6 = analogRead(S6);
  if (lS6 <= 500 && on == 1){
    digitalWrite(K13, HIGH);
    Serial.println("nivel de presión dos");
    delay(700);
    Tref = millis();
    while (lS6 <= 500 && on == 1){
      digitalWrite(K7, HIGH);
      lS6 = analogRead(S6);
      alarma(10000, 4);
      apagar();
    }
    digitalWrite(K7, LOW);
    digitalWrite(K13, LOW);
    if(on == 1){
      Serial.println("nivel de presión uno");
    }
  }
  if(on == 1){
    Serial.println("posición correcta malacate");
  }
}
void ciclos(){   //función para el numero de ciclos
  while(cont1 < cic && on ==1){   // siempre que el contador de ciclos sea menor
que la cantidad de ciclos requeridos hace...
```

```
    cont3++;     // suma una unidad a el contador de ciclos para la impresión
    Serial.print("ciclo número ");
    Serial.println(cont3);        //   imprime en número de ciclo
    act_4_mas();
    act_2_mas();
    act_1_mas();
    act_4_men();
    act_3_mas();
    act_2_men();
    act_1_men();
    act_3_men();

    if(cic != 0){ // si el número de ciclos es cero el ciclo se repetirá hasta
que se apague el motor
      cont1++;   //suma una unidad al número de ciclos
    }
  }
}
void act_1_mas(){   // función para cilindro 1( + )
  lS1 = analogRead(S1);
  Tref = millis();
  while (lS1 <= 500 && on == 1){
    digitalWrite(K0, HIGH);
    lS1 = analogRead(S1);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K0, LOW);
  fdelay(t1mas);
}
void act_1_men(){   // función para cilindro 1( - )
  lS0 = analogRead(S0);
  Tref = millis();
  while (lS0 <= 500 && on == 1){
    digitalWrite(K1, HIGH);
    lS0 = analogRead(S0);
    alarma(10000, 1);
    apagar();
  }
  digitalWrite(K1, LOW);
  fdelay(t1men);
}
void act_2_mas(){   // función para cilindro 2( + )
  lS3 = analogRead(S3);
  Tref = millis();
  while (lS3 <= 500 && on == 1){
    digitalWrite(K2, HIGH);
    lS3 = analogRead(S3);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K2, LOW);
  fdelay(t2mas);
}

void act_2_men(){   // función para cilindro 2( - )
  lS2 = analogRead(S2);
  Tref = millis();
  while (lS2 <= 500 && on == 1){
    digitalWrite(K3, HIGH);
```

```
    lS2 = analogRead(S2);
    alarma(10000, 2);
    apagar();
  }
  digitalWrite(K3, LOW);
  fdelay(t2men);
}
void act_3_mas(){   // función para cilindro 3( + )
  lS5 = analogRead(S5);
  Tref = millis();
  while (lS5 <= 500 && on == 1){
    digitalWrite(K4, HIGH);
    lS5 = analogRead(S5);
    alarma(50000, 3);
    apagar();
  }
  digitalWrite(K4, LOW);
  fdelay(t3mas);
}

void act_3_men(){   // función para cilindro 3( - )
  lS4 = analogRead(S4);
  Tref = millis();
  while (lS4 <= 500 && on == 1){
    digitalWrite(K5, HIGH);
    lS4 = analogRead(S4);
    alarma(50000, 3);
    apagar();
  }
  digitalWrite(K5, LOW);
  fdelay(t3men);
}
void act_4_mas(){   // función para malacate( + )
  lS7 = analogRead(S7);
  Tref = millis();
  while (lS7 <= 500 && on == 1){
    digitalWrite(K6, HIGH);
    lS7 = analogRead(S7);
    alarma(10000, 4);
    apagar();
  }
  fdelay(t4mas);
}

void act_4_men(){   // función para malacate( - )
  digitalWrite(K6, LOW);
  if(on == 1){
    digitalWrite(K13, HIGH);
    Serial.println("nivel de presión dos");
  }
  delay(500);

  lS6 = analogRead(S6);
  Tref = millis();
  while (lS6 <= 500 && on == 1){
    digitalWrite(K7, HIGH);
    lS6 = analogRead(S6);
    alarma(10000, 4);
    apagar();
  }
```

```
  digitalWrite(K7, LOW);
  digitalWrite(K13, LOW);
  if(on == 1){
    Serial.println("nivel de presión uno");
  }
  fdelay(t4men);
}
void apagar(){
  var3 = analogRead(S11);  // lee estado de señal y almacena en var3
  if (var3 >= 500){      // convierte señal en cero o uno
    var3=1;
  }
  else {
    var3=0;
  }
  delay(10);              //retraso de 10 milisegundos
  var4 = analogRead(S11);  // lee estado de señal y almacena en var4
  if (var4 >= 500){      // convierte señal en cero o uno
    var4=1;
  }
  else {
    var4=0;
  }
  if (var3 == var4){      // evita rebote de contacto
    if (var3 != senal_apag){        // selecciona cuando hay  cambio de la señal
      if (var3 == HIGH){
        digitalWrite(K14, LOW);
        if (on == 1){    // selecciona cuando se suelta el pulsador
          digitalWrite(K12, LOW);    // ventear
          digitalWrite(K13, LOW);    // ventear
          digitalWrite(K10, LOW);    // apaga motor
          digitalWrite(K11, LOW);
          Serial.println("motor apagado");   //imprime "motor apagado"
          on = 0;
        }
      }
      senal_apag = var3;
    }
  }
}
void fdelay(int s){
  int a = 0;
  while (a <= s*1000 && on == 1){
    a = a + 100;
    delay (100);
    apagar();
  }
}
void alarma(unsigned int Talarma, int act){
  int T = -(Tref - millis());
  if (abs(T) >= Talarma && on == 1){
    digitalWrite(K14, HIGH);
    digitalWrite(K12, LOW);
    digitalWrite(K13, LOW);
    delay(1000);
    digitalWrite(K10, LOW);
    digitalWrite(K11, LOW);
    on = 0;
    Serial.print("ALERTA: hay problemas en el actuador ");
    Serial.println(act);
```

```
    Serial.println("Se recomienda revisar las electroválvulas y los sensores de
dicho actuador");
  }
}
```

8.9.8 Evaluación para los estudiantes.

1. Elabore un programa utilizando el lenguaje Ladder para el PLC y en C++ para el Microcontrolador que permita realizar la secuencia indicada en el ítem de programación.
2. ¿Qué función cumple la tarjeta de interface de señales de entada?
3. ¿Qué función cumple la tarjeta de potencia?
4. ¿Cuál es la tensión de las señales de estada y salida de la tarjeta de control Arduino?

9. DISEÑO DEL MANUAL DE PROGRAMACIÓN DEL PLC Y MICROCONTROLADOR

Este manual brinda las herramientas básicas para la programación del PLC y el Microcontrolador. El manual consta de dos secciones principales:

- Programación del PLC DL06 DD1.
- Programación del Microcontrolador.

9.1 PROGRAMACIÓN DEL PLC DO DL06 DD1.

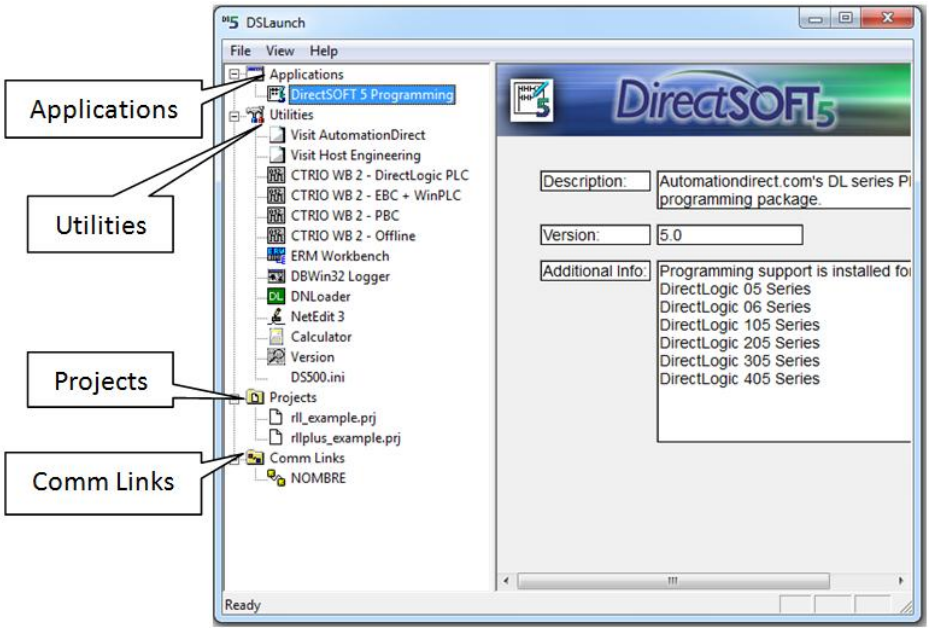
En esta sección se hace una breve descripción del programa DirecSoft, sus principales comandos (entradas, salidas, temporizadores, contactores, etc.), conexión entre el PLC con el dispositivo de programación (PC). También se exponen breves ejemplos de programación LADDER, el cual es el lenguaje que se utiliza en DirecSOFT para la programación del PLC DL06 DD1.

9.1.1 Descarga e instalación del software. La programación del equipo se realizar por medio del programa *DirectSOFT* V5, en su versión gratuita *PC-S100*, la cual permite programar hasta cien palabras. Para su instalación siga los siguientes pasos:

- Descargue esta versión en http://www.automationdirect.com, en el link *Search* escriba PC-DS100 y descargue.
- Ejecute el archivo tipo Setup.exe.

9.1.2 Familiarización con *DirectSOFT.* Abra el programa Observe en el lado izquierdo diferentes aplicaciones que el programa ofrece, como se observa en la figura 158.

Figura 158. Aplicaciones de *DirectSOFT.*



Fuente. Autores.

9.1.2.1 Applicatións. Se utiliza para crear un nuevo programa haciendo doble clic en *Direct*SOFT programming.

9.1.2.2 Utilities. Se pueden encontrar varias utilidades disponibles en esta carpeta, algunas se pueden comprar en AutomationDirect.

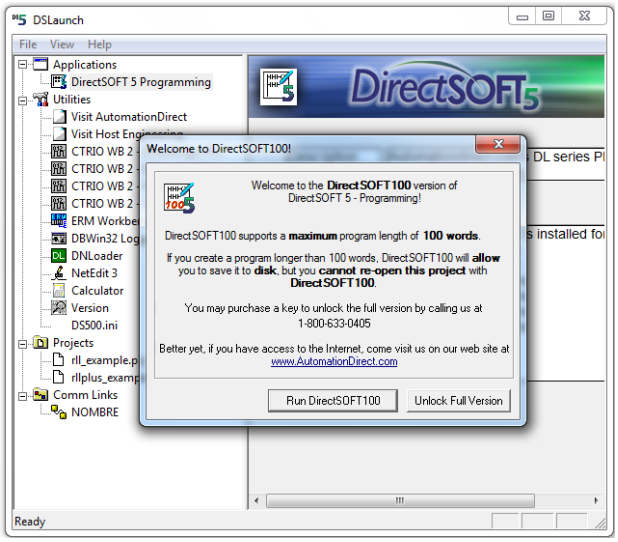
9.1.2.3 Projects. Esta carpeta guardará los proyectos que usted ha elaborado (siempre y cuando usted los haya salvado), los enumerará en el orden más recientemente utilizados.

9.1.2.4 Comm Links. Guardará los enlaces de comunicación que usted ha creado para establecer comunicación entre el PLC y su PC, note que el link tendrá en nombre que usted le ha asignado en el momento de añadir este link.

9.1.3 Comenzar a crear o modificar un programa

a) Abra el programa y pulse doble clic en *Direct*SOFT *Programming*, situado en la carpeta *Applications* del menú, posteriormente presione *RunDirecrsoft100.*tal como se muestra en la figura 159.

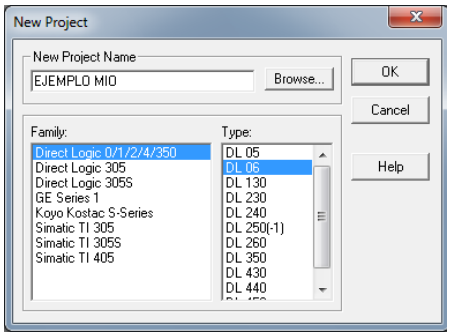
Figura 159. Iniciar el programa.



Fuente: Autores.

b) En la ventana *New Project*, llene el nombre del proyecto, seleccione en *Family* la opción *Direct Logic 0/1/2/4/350*, luego seleccione en *Type* la opción *DL06* y presione *OK* (ver figura 160).

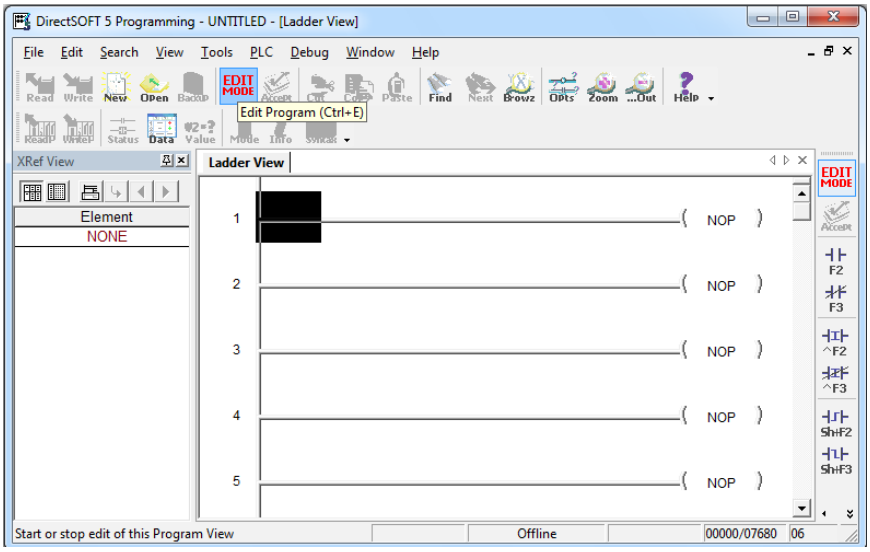
Figura 160. Ventana New Proyect.



Fuente: Autores.

c) Valla al modo *EDIT* para comenzar a escribir modificar un programa, haciendo clic sobre el botón *Edit Mode* en la barra de herramientas como se observa en la figura 161, otra manera de entrar a este modo es apretar las teclas Ctrl + E simultáneamente. Note que inmediatamente se activan los comandos de la herramienta *LADDER PALETTE* situados al lado derecho de la ventana, entre ellos están *F2* "contacto normalmente abierto", también observe que se torna de color negro el lugar donde se desea colocar un comando.

Figura 161. Ventana de programación.



Fuente: Autores.

9.1.3.1 Programar entradas del tipo X. Se programará un contacto normalmente abierto en el renglón 1, este será la entrada *X0* del PLC, pulse el icono de normalmente abierto o *F2* en la barra de *LADDER PALETTE*, verá inmediatamente una ventana con el cursor de texto que por defecto indicará *C0*, también verá un indicador de color verde, este significa que dirección es correcta como se aprecia en la figura 162.

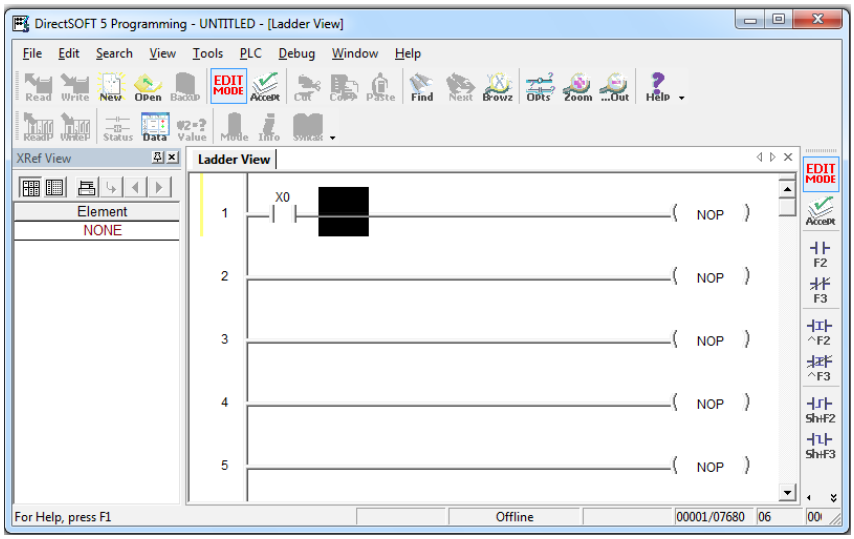
En caso de colocar una dirección inválida, la letra *O* en vez del digito *0* el indicador se tornará de color rojo que significa que hay un error de escritura. Cambie *C0* por *X0*, verifique que indicador está verde y luego presione la marca de verificación o *Enter*.

Figura 162. Ventana del elemento X0.

Fuente: Autores

El cursor se ha movido a la siguiente posición, usted podrá colocar otra función en este espacio (ver figura 163).

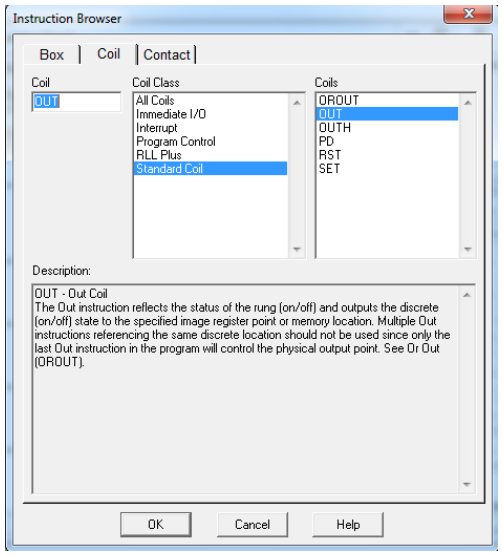
Figura 163. Ejemplo de entrada tipo x.



Fuente: Autores

9.1.3.2 Programar salidas del tipo Y. Sitúe el cursor en la instrucción *NOP*, haga clic en el botón *Coils* (bobinas) en *Tool Palette*. El *instruction browser* aparecerá con la bobina estándar *OUT* seleccionada por defecto como se observa en la figura 164. Haga clic en *OK* para entrar una bobina estándar.

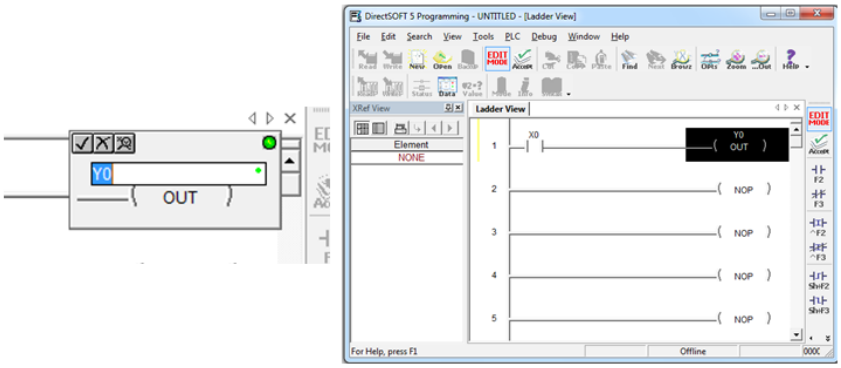
Figura 164. Ventana de Instruction Browser seleccionado la salida OUT.



Fuente: Autores

Nuevamente por defecto aparecerá *C0*, digite *Y0* y luego presione *Enter* (ver figura 165*).*

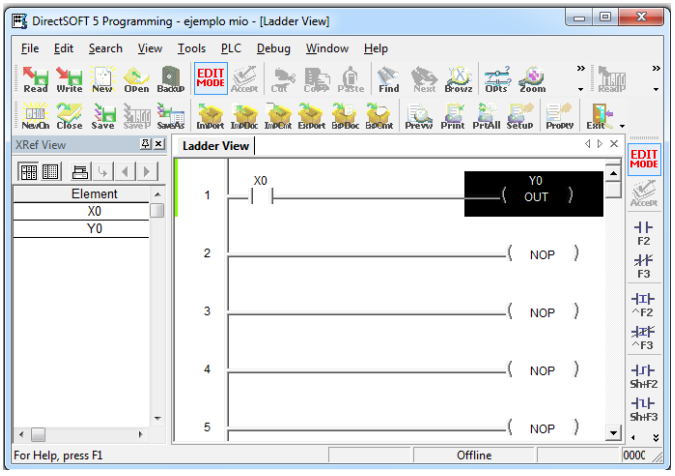
Figura 165. Ventana del elemento Y0.



Fuente: Autores.

Se puede observar en figura 165 la una barra amarilla a lado del renglón esta indica que se ha entrado una instrucción o instrucciones, pero que el programa no se ha aceptado (compilado), para aceptar el programa usted deberá haber completado el renglón, luego pulse *AccePt*. Los renglones que se han acepta cambiarán a color verde. Para ver el icono *Save* (Salvar en el disco) sitúese sobre el icono *AccePt* luego clic derecho y active *File Bar*, ahora puede salvar el programa (ver figura 166).

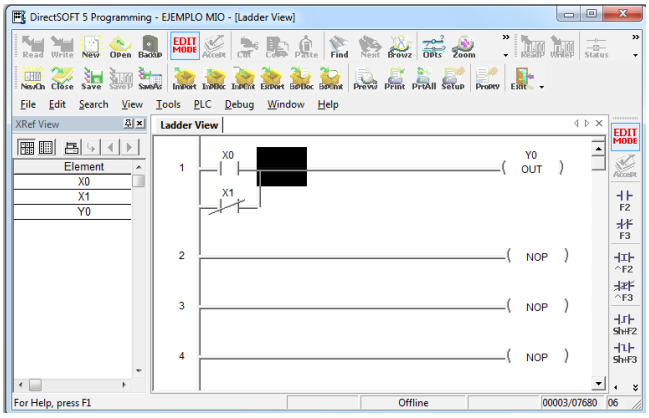
Figura 166. Ejemplo de entrada tipo Y.



Fuente: Autores.

9.1.3.3 Agregar contacto en paralelo. Para agregar un contacto en paralelo con el primero pulse la tecla *Enter* y se abrirá un espacio debajo de este renglón (ver figura 167), allí usted puede colocar otro contacto, en este caso se programará uno normalmente cerrado (F3) llamado *X1*, de la misma manera que en el ejemplo 4.4. Luego cierre el circuito usando las teclas CTRL + flecha para arriba simultáneamente. Acaba de programarse el renglón 1 usted puede programar cuantos reglones sean necesarios para la ejecución de su programa. Pulse el botón *Accept*, y salve.

Figura 167. Ejemplo de contacto en paralelo.

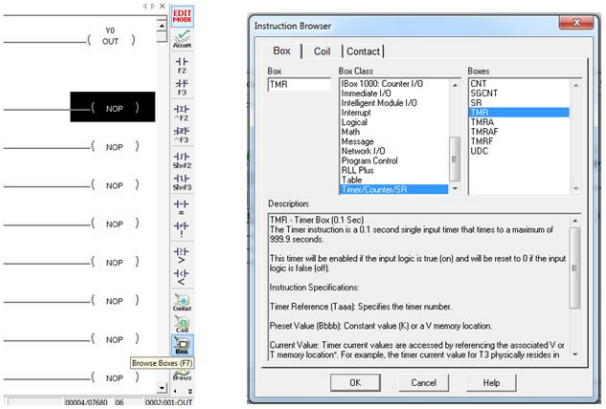


Fuente: Autores.

9.1.3.4 Programar un temporizador. Se programará un temporizador de 10 segundos en el renglón 2 que será activado por la entrada *X3*, para entrar *X3* siga las instrucciones en el inciso *"programando entradas del tipo X"*

Mueva el cursor sobre la instrucción *NOP* en el renglón 2 como se aprecia en la figura 168, y haga clic en el botón *Box* en *Ladder Pallete*, aparecerá la ventana *Instruction Browser*, en la sección de *BOX CLASS* (clase de box) seleccione *Timer/Counter/SR*, luego en la sección *Boxes* seleccione *TMR*, posteriormente puse *OK*.

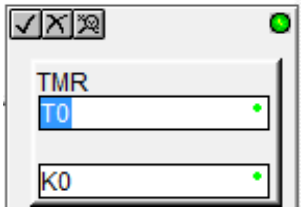
Figura 168. Ventana de Instruction Browser seleccionado en temporizador TMR.



Fuente: Autores.

Aparecerá la ventana de la figura 169, *TMR* es la identificación del temporizador en este caso T0, y *K* es el tiempo, cada unidad corresponde a 0,1 segundos, de modo que para programar 10 segundos se debe escribir *K100*, pulse *Enter.*

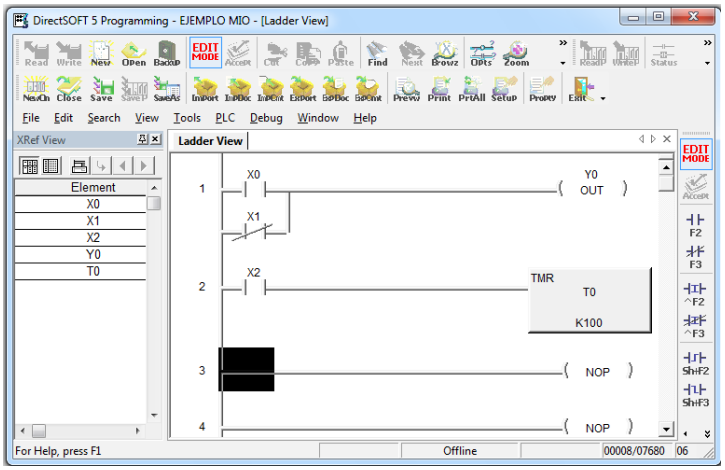
Figura 169. Ventana del temporizador.



Fuente: Autores.

Ya está agregada la salida del renglón que activará la memoria, pero aun hace falta el contacto que será activado por la memoria , se programará uno normalmente cerrado en el renglón 3, de tal forma que después de 10 segundos este se abrirá, presione clic sobre el renglón 3 (ver figura 170).

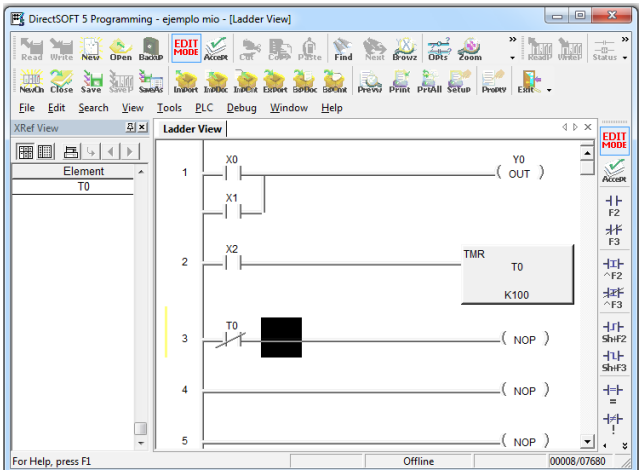
Figura 170. Programando temporizador.



Fuente: Autores.

Seleccione un contacto normalmente abierto (*F2*) en *Tool Palette*, nuevamente aparece *C0* por defecto, escriba *T0* y pulse *Enter* (ver figura 171).

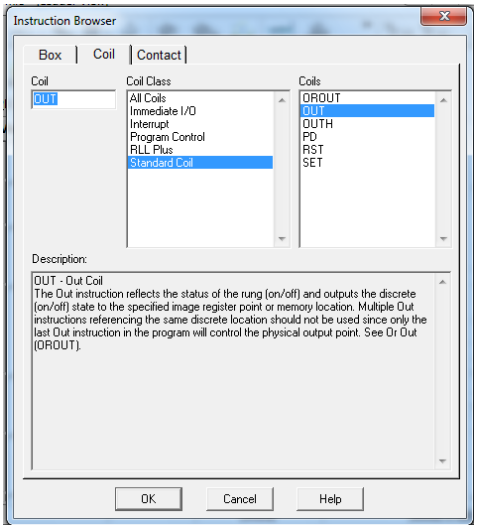
Figura 171. Ejemplo de temporizador.



Fuente: Autores.

9.1.3.5 Programar una memoria. Se programará una memoria en el renglón 3 activada por el temporizador T0, para ello, esto se realiza de igual manera que las salidas tipo Y. Sitúe el cursor en la instrucción *NOP*, haga clic en el botón *Coils* (bobinas) en *Tool Palette* como se muestra en la figura 172, en *instruction browser* aparecerá la bobina estándar (OUT) seleccionada por defecto, haga clic en OK para entrar una bobina estándar.

Figura 172. Ventana de Instruction Browser seleccionado la bobina OUT.

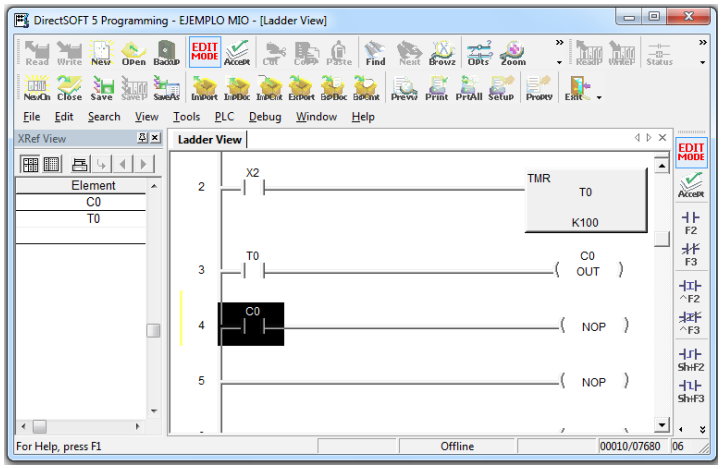


Fuente: Autores.

Por defecto aparece la memoria C0, usted puede darle la numeración que deseé, en este caso se dejará C0.

Ahora como en el ejemplo del temporizador hace falta colocar el contacto que será activado por esta memoria, este se programará en el renglón 4 de la misma manera que en el ejemplo del temporizador, mueva el cursor sobre el renglón 4 y seleccione un contacto normalmente cerrado (F2), escriba C0 y pulse Enter. Ver figura 173.

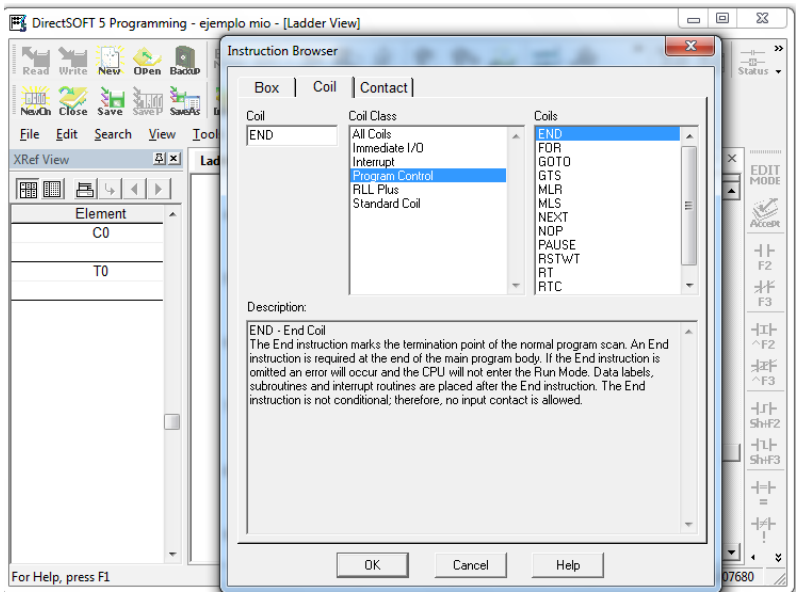
Figura 173. Ejemplo de una memoria.



Fuente: Autores.

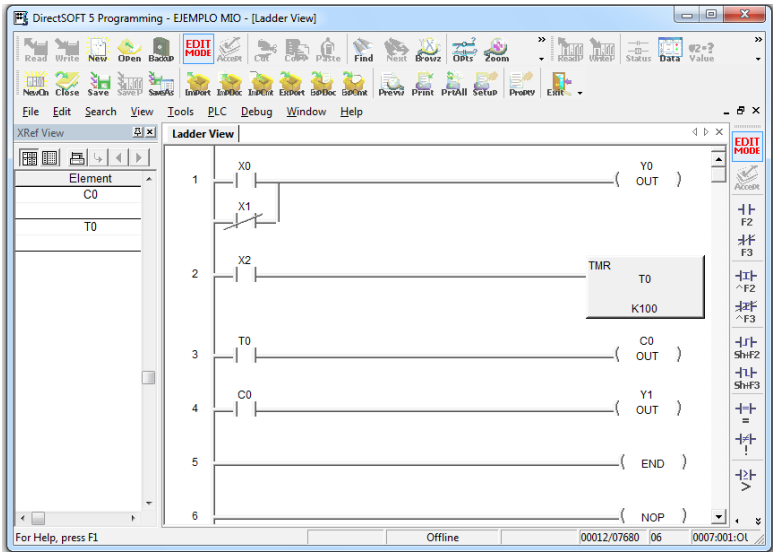
9.1.3.6 Entre la bobina END. Todo programa debe terminar en la bobina *END*. Para programar este renglón mueva el cursor hasta *NOP* en el último renglón del programa, en este caso en el 5, haga clic en el botón *COIL* (bobina), aparecerá la ventana *Instruction Browser* (ver figura 174), en la sección *Coil Class*, seleccione *Program Control*, luego en la sección *Coils* seleccione la bobina *END*, presione *OK*, aparecerá nuevamente la ventana de confirmación, presione *Enter*, salve, y ya está terminado el programa (ver figura 175)

Figura 174. Ventana de Instruction Browser seleccionado la instrucción END.



Fuente: Autores.

Figura 175. Ejemplo de programación de la bobina END.

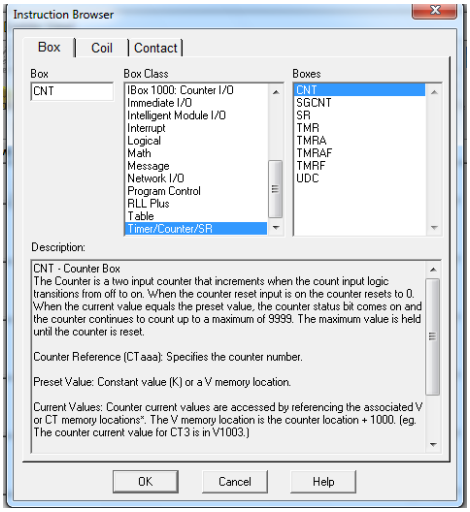


Fuente: Autores.

9.1.3.7 Contador CNT. El programa del PLC una vez barre la totalidad de los renglones, se ejecuta de manera repetitiva, hasta que se pase del modo *RUN* al modo *STOP*, al momento de hacer esto, si se está controlando un cilindro, este podría quedar a mitad de camino. Lo ideal es que el programa se detenga en el momento exacto de haber terminado la instrucción *END*. Para controlar esto, se pueden programar el número de veces que se ejecute el programa, para ello se puede utilizar un contador *CNT*, este contador consta de dos entradas, la primera sirve para contar las veces que se ha ejecutado el programa, y la segunda es para resetearlo.

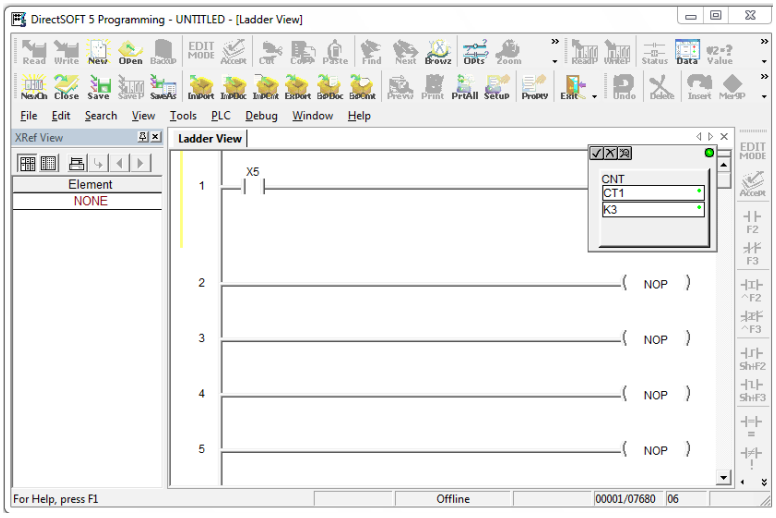
Para programa un contador el cursor hasta *NOP*, y haga clic en el botón *Box* en *Ladder Pallete*, aparecerá la ventana *Instruction Browser*, en la sección de *BOX CLASS* (clase de box) ver Figura 176, seleccione *Timer/Counter/SR*, luego en la sección *Boxes* seleccione CNT. En la ventana emergente, la letra K va acompañada del número de veces que se desea que corra el programa, es decir si se desea que el programa corra tres veces se debe programar *K3* (ver figura 177).

Figura 176. Ventana de Instruction Browser seleccionando un contador CNT.



Fuente: Autores.

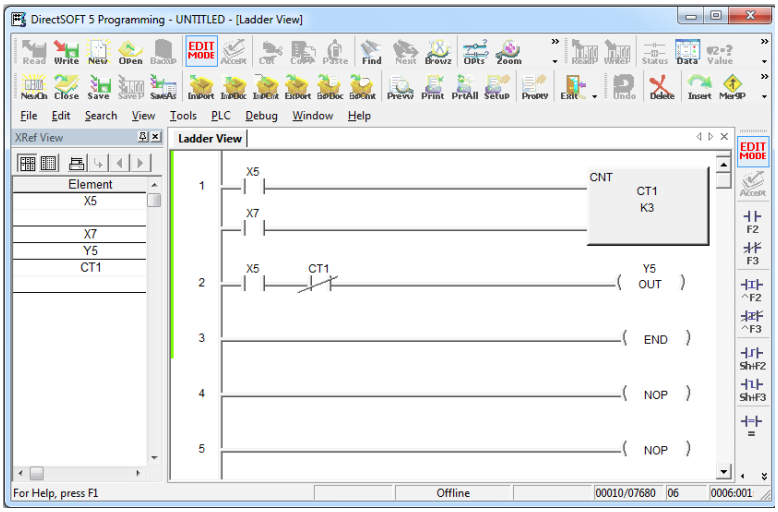
Figura 177.Programando un contador CNT.



Fuente: Autores.

En el ejemplo siguiente, cuando *X5* hace una transición de OFF para ON, el valor corriente del contador *CT1* se incrementará en uno, como se programó *K3*, el programa correrá tres veces. Cuándo la entrada *RESET X7* es activada, el contador se vuelve a cero y el programa puede correr nuevamente 3 veces (ver figura 178).

Figura 178. Ejemplo de programación de un contador CNT.



Fuente: Autores.

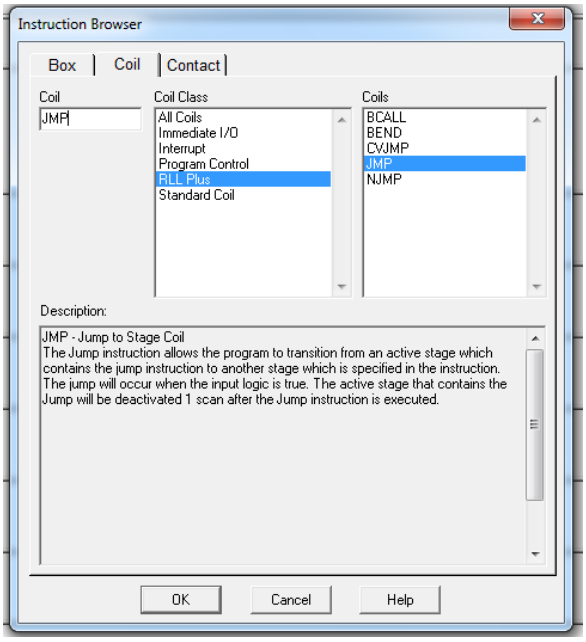
9.1.3.8 Programación por etapas. Mucha veces se desea que una función se ejecute antes que otra, por ejemplo, para comenzar a realizar una secuencia con un actuador, este debe estar en su posición inicial, si no es así, entonces se necesita que antes de realizar la secuencia el actuador sea llevado a esta posición, es decir se necesita ejecutar una etapa de comprobación de posición del actuador y posteriormente la secuencia que se desea que el mismo realice, una de las muchas formas de hacer esto es mediante la programación por etapas, la cual garantiza que se cumplan primero ciertas funciones, para posteriormente ejecutar las siguientes funciones.

En el programa DirectSOFT, una etapa es representada por la instrucción SG. A continuación se dará un ejemplo:

Se tiene un actuador el cual posee un sensor al inicio (X1) y al final de carrea (X2), se desea que el actuador se mueva hacia la segunda posición por medio de Y1 y se regrese por medio de Y2, el actuador solo es activado por sus sensores, si estos no lo detectan el actuador no se moverá. Entonces en un evento en el cual el actuador no esté en su posición inicial el programa debe ser capaz de ubicarlo en esta. Para lograr ello hacemos lo siguiente:

Inicialmente, se enciente el motor con el contacto X1 y la salida Y1, paralela a la salida Y1 se programará una salida JMP para que active la etapa de comprobación, en COIL escriba JMP y presione OK (Ver figura 179).

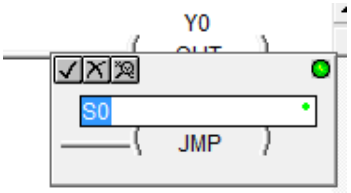
Figura 179. Ventana de Instruction Browser seleccionando una bobina JMP.



Fuente: Autores.

Escriba el nombre de la etapa, en este caso S0, y presione enter, como se observa en la figura 180.

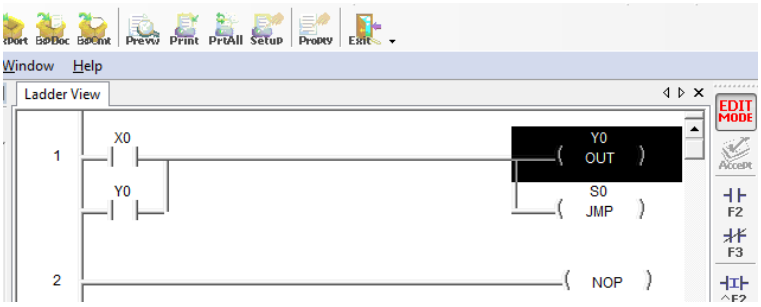
Figura 180.Ventana del elemento JMP.



Fuente: Autores.

Posteriormente y al mismo tiempo presione las teclas Ctrl más▲, para unir la salida con el renglón 1. Acepte y guarde (Ver figura 181).

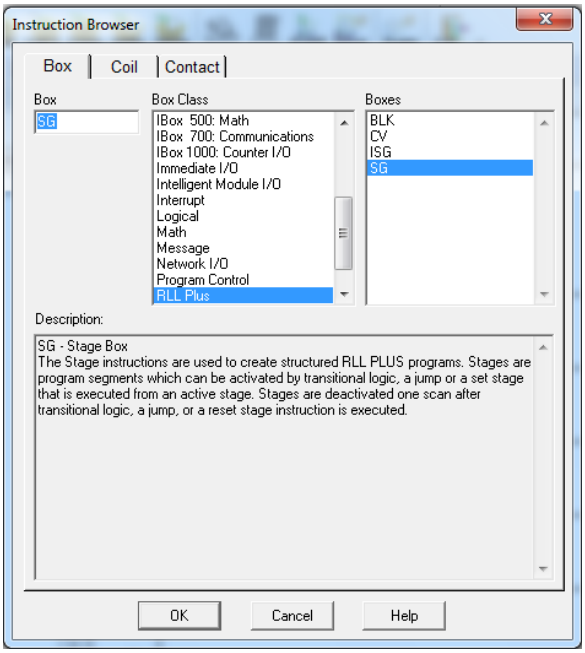
Figura 181. Programación del renglón 1 del ejemplo de programación por etapas.



Fuente: Autores.

Posicione el cursor sobre el renglón 2, Abra BOX, escriba SG, y presione OK (ver figura 182).

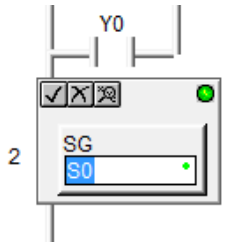
Figura 182. Ventana de Instruction Browser seleccionando un elemento SG.



Fuente: Autores.

Escriba el nombre de la etapa, para este caso S0, y presione enter, como se observa en la figura 183.

Figura 183. Ventana del elemento SG.

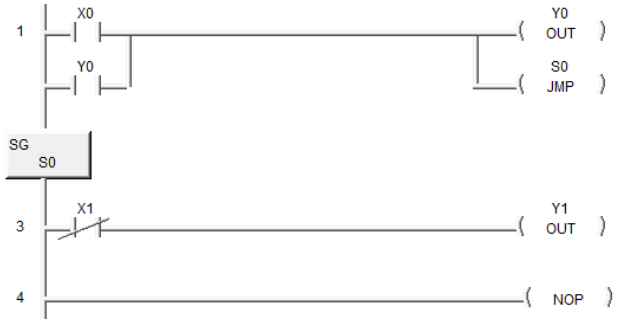


Fuente: Autores.

Realizado lo anterior, ya se ha programado el inicio de una etapa, ahora se procede a programar la comprobación de la posición del cilindro, y en su defecto corregirla.

Se programa un contacto normalmente cerrado, y la salida Y1, de tal forma que si inicialmente el actuador no está en la posición inicial, se activa la salida Y1, para ubicarlo, esto se puede observar en la figura 184.

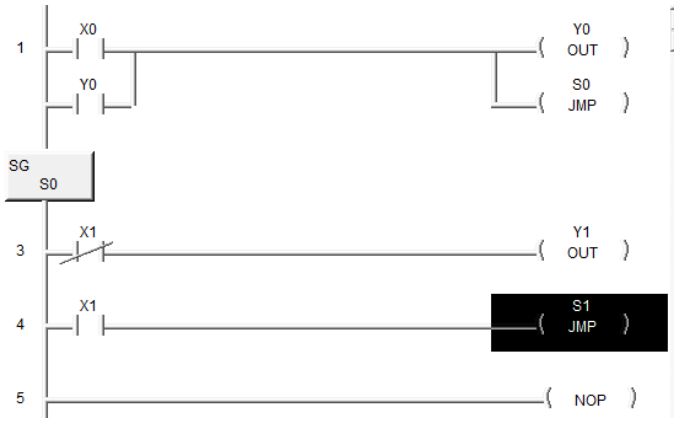
Figura 184. Programación del renglón 3 del ejemplo de programación por etapas.



Fuente: Autores.

Una vez sensor detecta que la posición es correcta, este activa la siguiente etapa, para ellos se programa una salida JMP de igual forma que en el renglón 1, ahora con el nombre de la siguiente etapa, para este caso S1. En la figura 185 se puede observar esta programación.

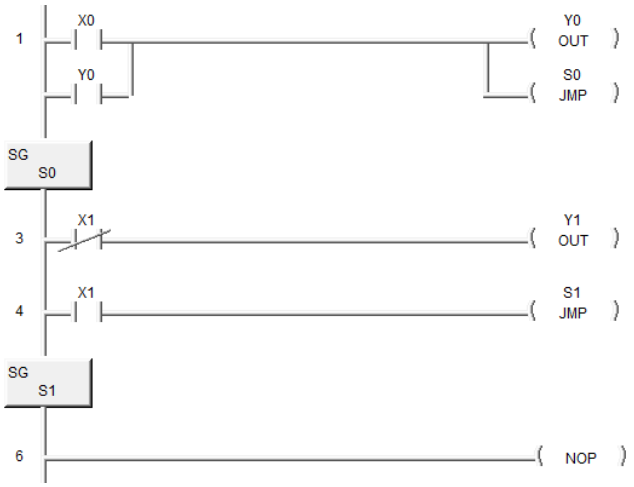
Figura 185. Programación del renglón 4 del ejemplo de programación por etapas.



Fuente: Autores.

Se programa una nueva etapa SG llamada S1, esta va a ser la etapa que ejecutará la secuencia deseada (Ver figura 186).

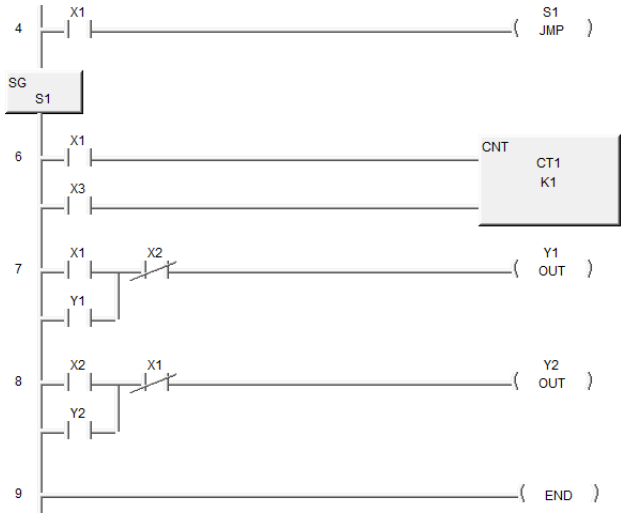
Figura 186. Programación del renglón 5 del ejemplo de programación por etapas.



Fuente: Autores.

Ahora se programa la secuencia deseada, y por ultimo END como se aprecia en la figura 187, para culminar el programa. También se puede programar un contador CNT con su reset X3, para que realice la secuencia solo una vez.

Figura 187. Ejemplo de programación por etapas.



Fuente: Autores.

9.1.3.9 Documentación. La documentación se refiere a los comentarios en renglones, apodos del elemento, descripciones de un elemento, etc., que son necesarios para hacer el programa más claro para cualquier persona que pueda para leerlo en el futuro estos se pueden visualizar en la figura 188. La documentación se puede agregar a un programa en cualquier momento, pero es recomendado, si está modificando el programa, tener la práctica de incluirlos en el programa. Esta información solo queda en el disco duro, no en el PLC.

La mayoría de la documentación se refiere a elementos individuales y es por lo tanto específico en naturaleza. Se enumeran abajo cinco tipos de documentación usados en DirectSOFT.

Elementos. Direcciones de los elementos, es decir, X1, Y10, etc.

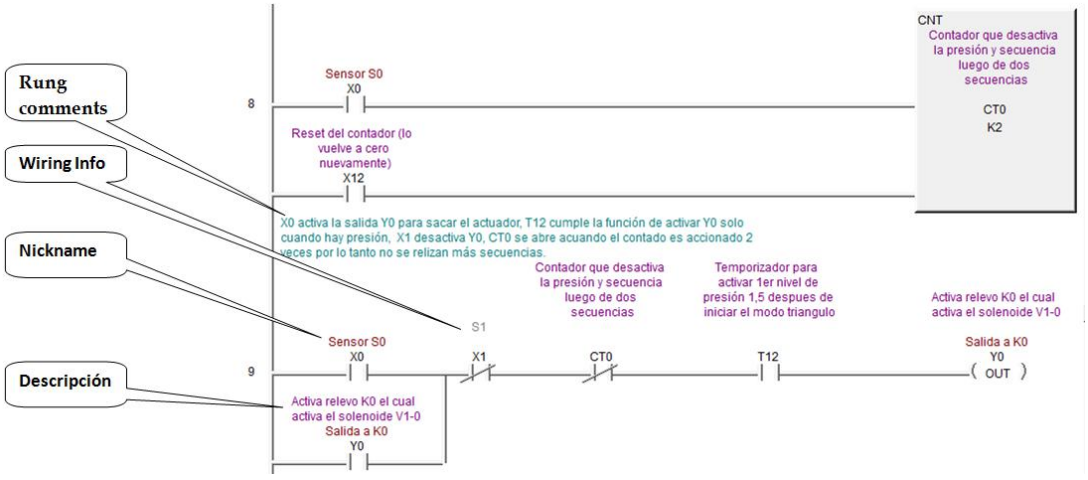
Nicknames (Apodos). Nombres alfanuméricos se pueden utilizar para los varios tipos de elementos de programa. Es generalmente más fácil recordar el nombre botón partir que X1 es el punto de la entrada para el botón.

Descriptions (Descripciones). Descripciones más largas del elemento. Usted puede también utilizar esta área para incluir breves pasos de búsqueda de problemas.

Wiring information (Información del cableado). La información del cableado puede ayudarle rápidamente a identificar el cableado del panel para un punto específico. Por ejemplo, usted puede saber que X1 es botón partir, pero usted tiene que encontrar generalmente otro documento para saber qué número de cable está conectando la entrada para comenzar la búsqueda de fallas.

Rung comments (comentarios). Los comentarios del renglón se asignan a un renglón individual.

Figura 188. Tipos de documentación.



Fuente: Autores.

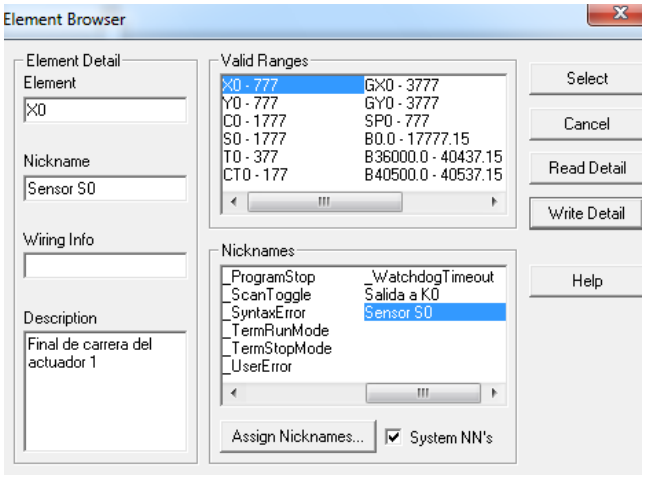
Para escribir *Nicknames, Descriptions, Wiring information,* de cada elemento, en su respectivo cuadro de asignación pulse sobre el tercer link, como lo indica la flecha roja en la figura 189.

Figura 189. Ventana del elemento X0.

Fuente: Autores.

Aparecerá la ventana de *Element Detail*, escriba en la documentación que desea, Presione *Write Detail*, y posterior mente, *Select*, (ver figura 190).

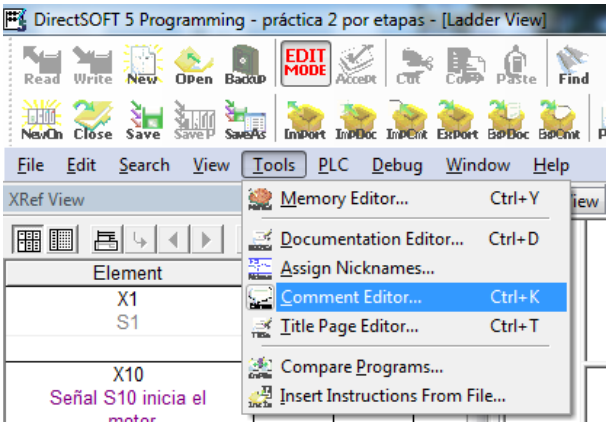
Figura 190. Ventana de Element Detail.



Fuente: Autores.

Para escribir los *Rung comments*, presione sobre el link Tools como lo indica la figura 191, y posteriormente en *Comment Editor*.

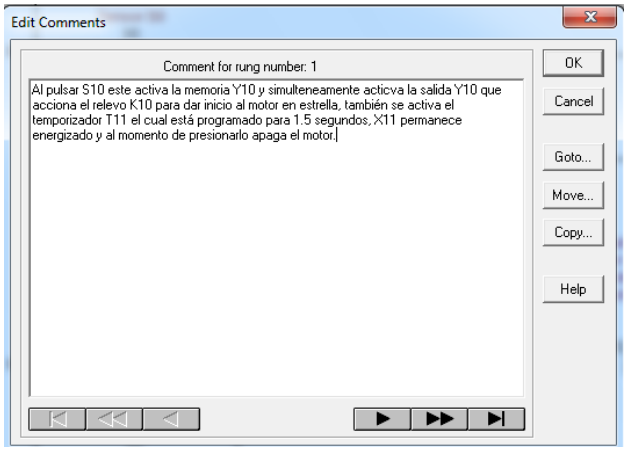
Figura 191. Ventana de Tools.



Fuente: Autores.

Aparecerá la ventana *Edit Comments,* Escriba el comentario de cada renglón y presione OK (Ver figura 192).

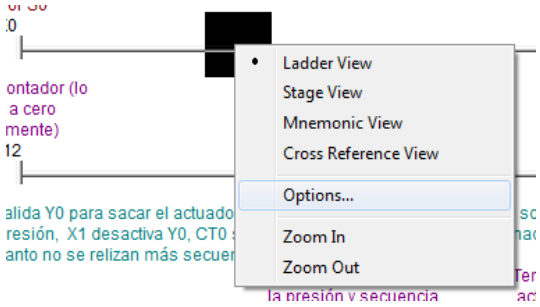
Figura 192. Ventana Edit Comments.



Fuente: Autores.

Para ver la documentación, presione Clic, derecho sobre cualquier renglón y aparecerá la ventana con varios link entre ellos *Options* como se observa en la figura 193 y Haga clic sobre este.

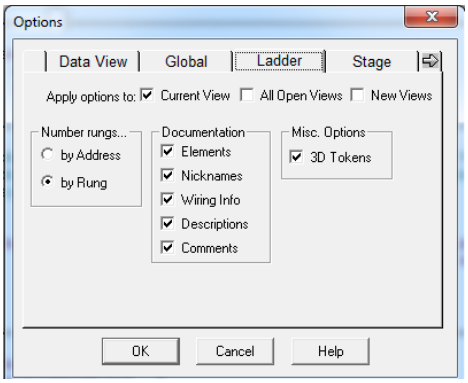
Figura 193 Ventana de menú.



Fuente: Autores.

Saldrá la ventana de *Options* (ver figura 194), en esta se puede seleccionar la documentación que se desea que aparezca, luego presione *OK.*

Figura 194. Ventana de Options.

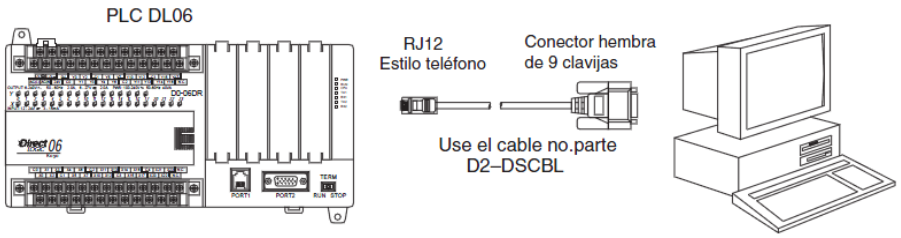


Fuente: Autores.

9.1.3.10 Conectar el PLC a la PC. El PLC debe estar encendido para realizar la conexión, de lo contrario no habrá conexión y cada vez que desee instaurar comunicación con PLC debe usar el mismo puerto USB.

Instale el drivers adaptador DB9-USB. El cable del cual dispone para instaurar comunicación entre el PLC el dispositivo de programación (en este caso su PC) es RJ12 estilo teléfono y conector hembra BD9 de 9 clavijas, por que se necesita un adaptador BD9 a USB como se observa en la figura 195.

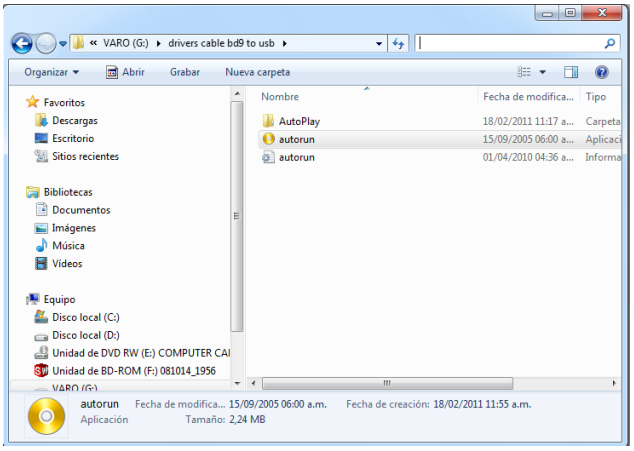
Figura 195. Cable de conexión entre el PLC y la PC



Fuente: Autores.

a) Inserte el DC o memoria USB contenedor del drivers ejecute el archivo autorun.exe que se aprecia en la figura 196.

Figura 196. Paso "a" instalación de los drivers.



Fuente: Autores.

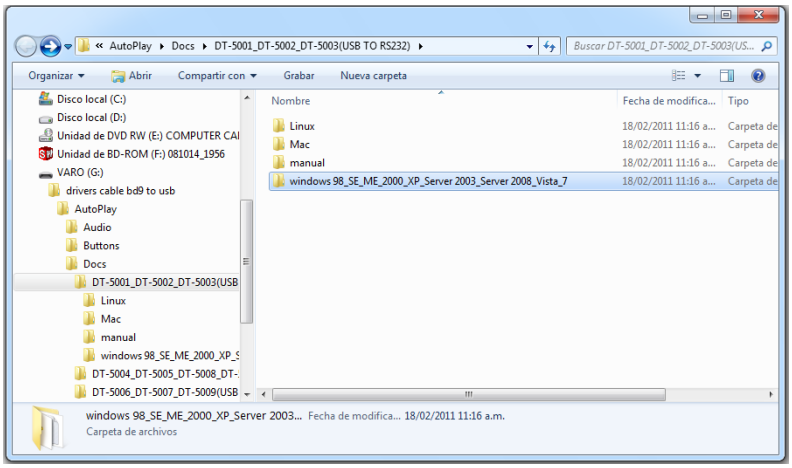
b) Presione doble clic a DT-5001 como se aprecia en la figura 197.

Figura 197. Paso "b" instalación de los drivers.

Fuente: Autores.

c) Abra la carpeta de Windows (ver figura 198).

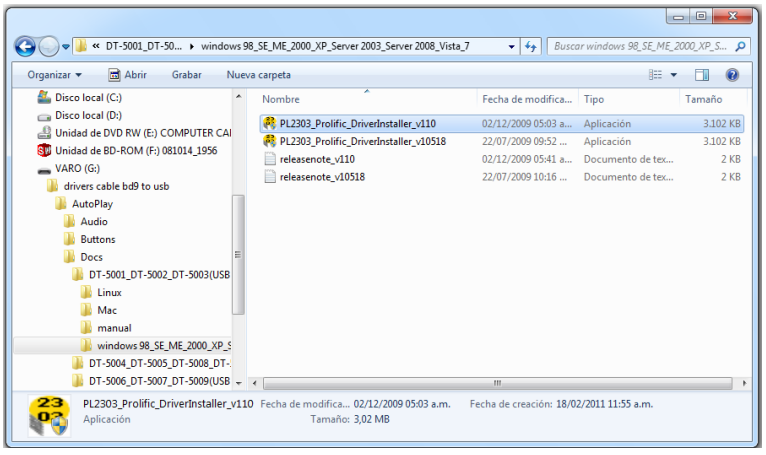
Figura 198. Paso "c" instalación de los drivers.



Fuente: Autores.

d) Ejecute cualquiera de las dos aplicaciones PL2303 (ver figura 199).

Figura 199. Paso "d" instalación de los drivers.

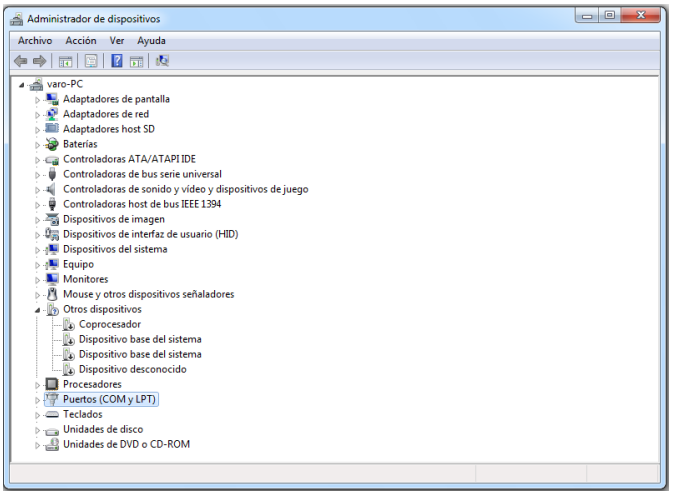


Fuente: Autores.

e) Inserte el cable en cualquiera de los puertos USB y diríjase a administrador de dispositivos, espere que el PC reconozca la conexión.

f) Al aparecer *Puertos (COM y LPT)* como se observa en la figura 200, ya está instalado.

Figura 200. Paso "f" instalación de los drivers.



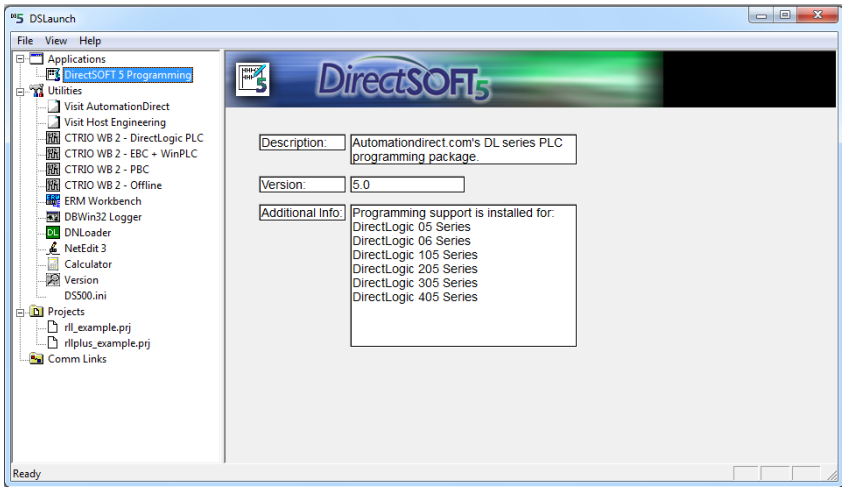
Fuente: Autores.

g) Instaurar comunicación entre el PLC y el PC. Diríjase al acceso directo que aparecerá de manera automática en el escritorio de su PC (ver figura 201), después de haber instalado el programa DirectSOFT V5 y ábralo, aparecerá la ventana que se observa en la (figura 202).

Figura 201. Icono de acceso directo de DirectSOFT.

Fuente: Autores.

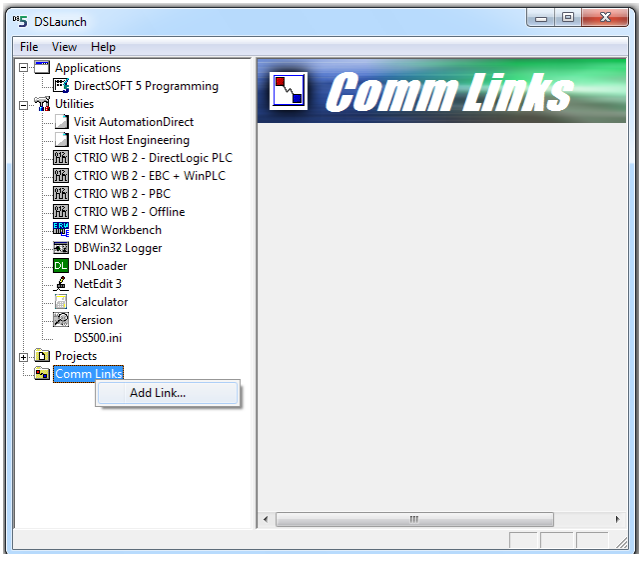
Figura 202. Ventana de inicio de DirectSOFT.



Fuente: Autores.

h) En *Comm Link* presione clic derecho luego clic en *Add Link* (ver figura 20).

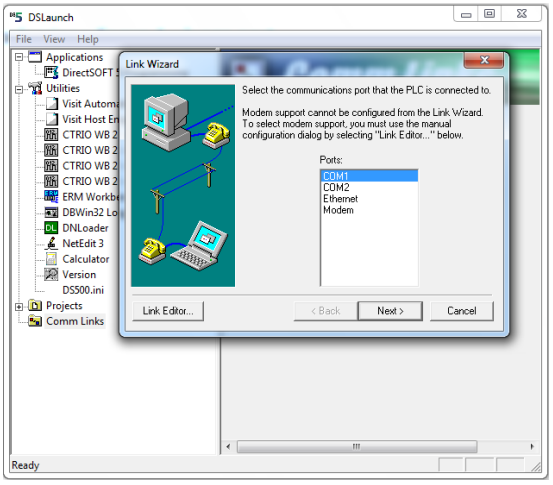
Figura 203. Ventana de Comm Link.



Fuente: Autores.

i) Seleccione *COM1* y presione en *Next* como se observa en la figura 204.

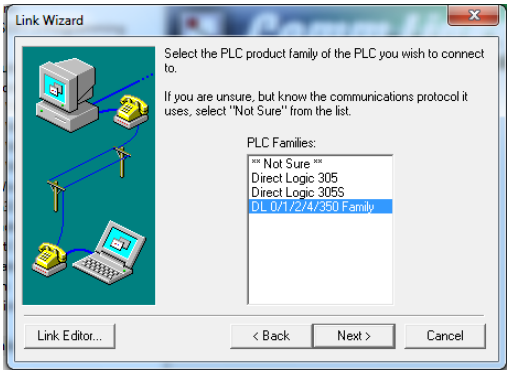
Figura 204. Ventana de link Winzard, opción selección de Ports.



Fuente: Autores.

j) Seleccione *DL 0/1/2/4/350 Falmily* y presione *Next*, como se observa en la figura 205.

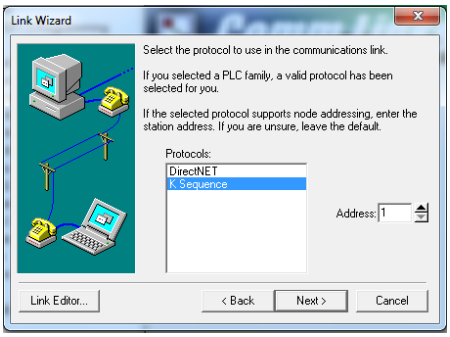
Figura 205. Ventana de link Winzard, opción selección PLC Families.



Fuente: Autores.

k) Seleccione *K Sequence* y en *Address* seleccione *1* luego presione *Next* (ver figura 206).

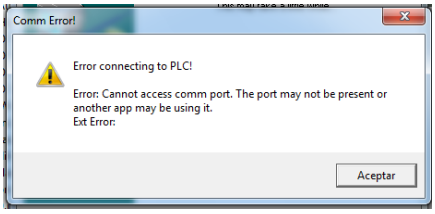
Figura 206. Ventana de link Winzard, opción selección de Protocols.



Fuente: Autores.

l) Espere, en caso de haber un error en la comunicación aparecerá la ventana que se observa en la figura 207.

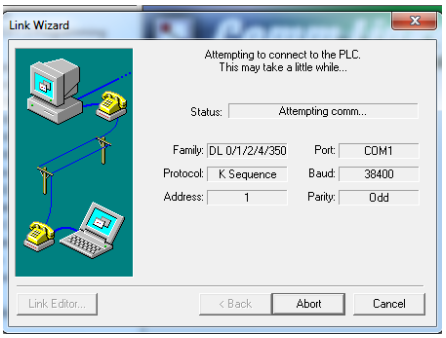
Figura 207. Ventana de error de comunicación.



Fuente: Autores.

Para ello diríjase al inciso q "*En caso error*", en caso de no haber error aparecerá la ventana de la figura 208.

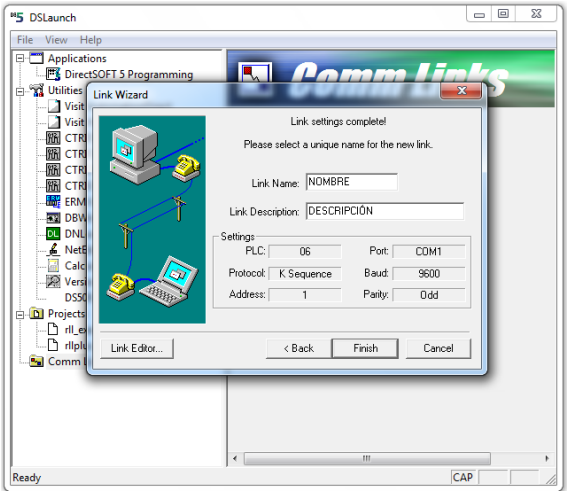
Figura 208. Ventana de reconocimiento del enlace.



Fuente: Autores.

m) Llene el link *Name* y el link *Description* y presione *Finish* (ver figura 209).

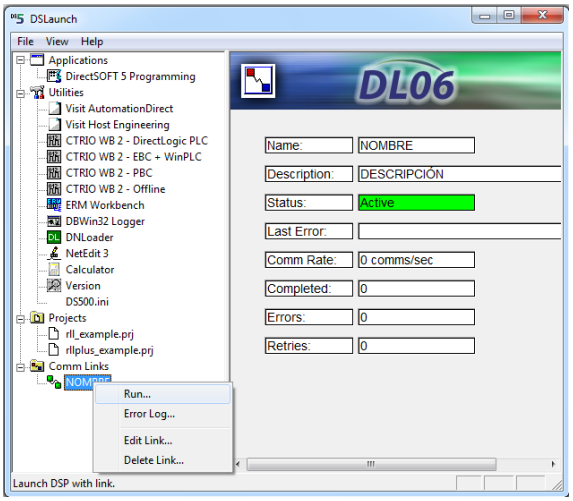
Figura 209. Ventana de referenciado del enlace



Fuente: Autores.

n) Abra la subcarpeta en *Comm Link* la cual es el enlace que usted creó (ver figura 210), en este caso *Nombre,* presione clic derecho y luego *Run*.

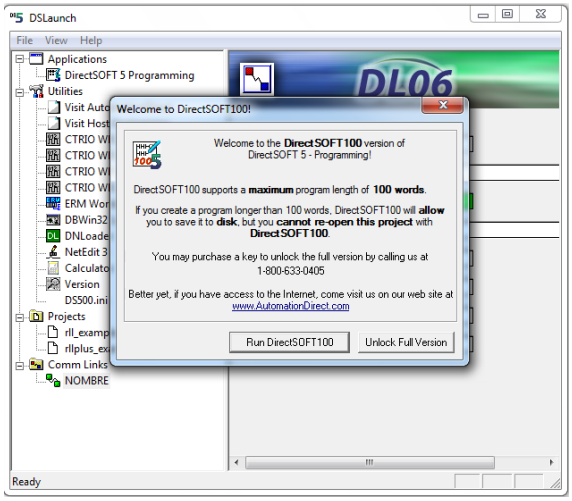
Figura 210. Ventana de enlace activado.



Fuente: Autores.

o) Presione el link *Run DirectSot100*, como se observa en la figura 211.

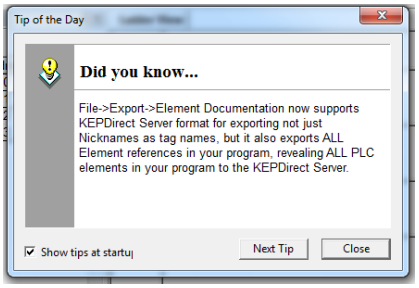
Figura 211. Ventana para entrar al programador.



Fuente: Autores.

p) Presione *Close (*ver figura 212*)*.

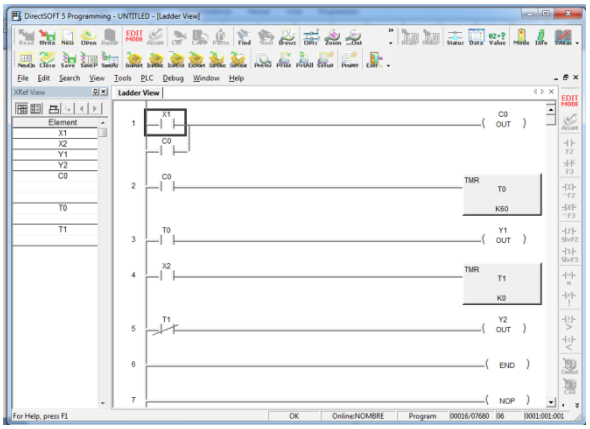
Figura 212. Ventana Tip of the Day.



Fuente: Autores.

Usted podrá visualizar el programa que está guardado en la memoria el PLC como se observa en la figura 213.

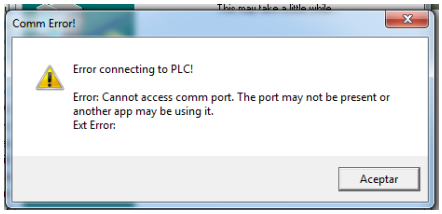
Figura 213. Ventana de programación.



Fuente: Autores.

q) En caso error como muestra la figura 214, realice lo siguiente:

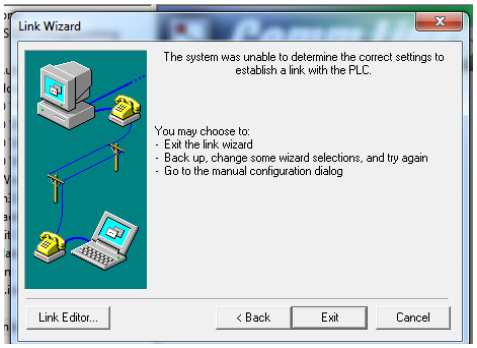
Figura 214. Ventana de error.



Fuente: Autores.

r) Presione *aceptar* y luego *back* (ver figura 215).

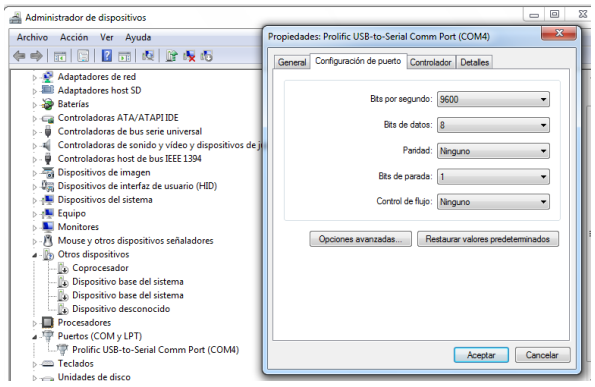
Figura 215. Ventana link Editor.



Fuente: Autores.

s) Abra en panel de control de su PC y valla a administrador de dispositivos, valla a *Puertos (COM y LPT)* abra la subcarpeta *Prolific USB-to-Serial Comm Port (COM4 en este caso)*, "*COM* puede estar acompañado por otro número" como se observa en la figura 216, presione clic derecho luego propiedades y posteriormente configuración de puerto.

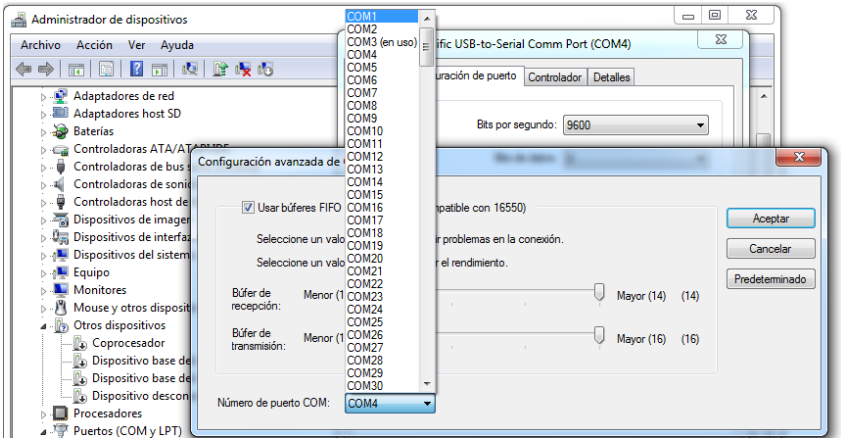
Figura 216. Propiedades del puerto de comunicación.



Fuente: Autores.

t) Presione opciones avanzadas y seleccione un puerto que no esté ocupado del 1 al 2 si su PC tiene dos entradas USB o del 1 al 4 si su PC tiene 4 entradas USB (ver figura 217).

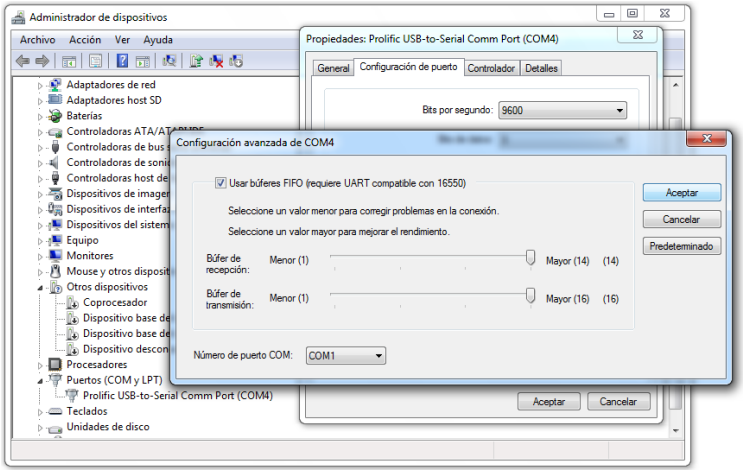
Figura 217. Selección del puerto comunicación.



Fuente: Autores.

u) Presione *aceptar*, en la ventana de la figura 218.

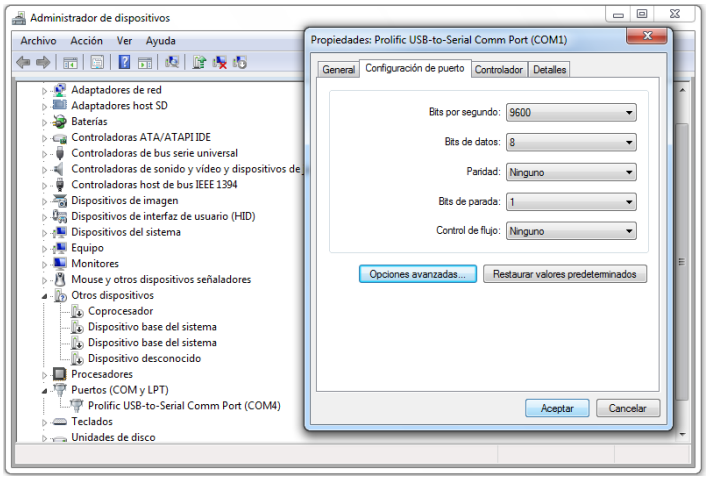
Figura 218. Configuración avanzada del puerto de comunicación.



Fuente: Autores.

v) Presione nuevamente *aceptar* (ver figura 219).

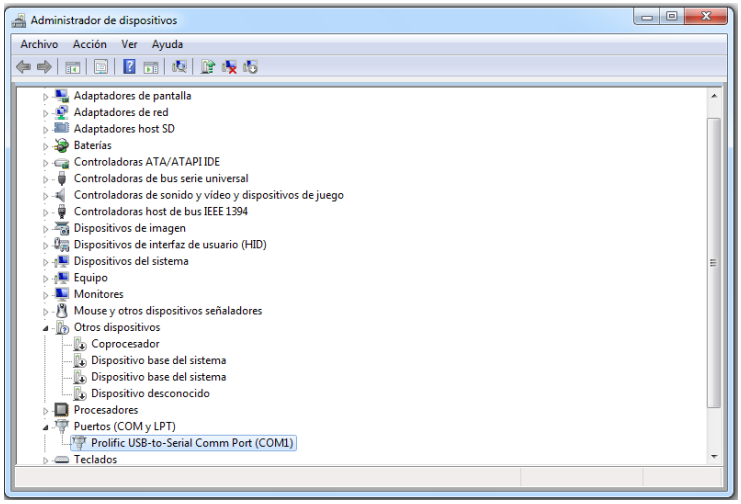
Figura 219. Configuración del puerto de comunicación.



Fuente: Autores.

Note en la figura 220, que cambió de (*Comm4)* a (*Comm1).*

Figura 220. Administrador de dispositivos.



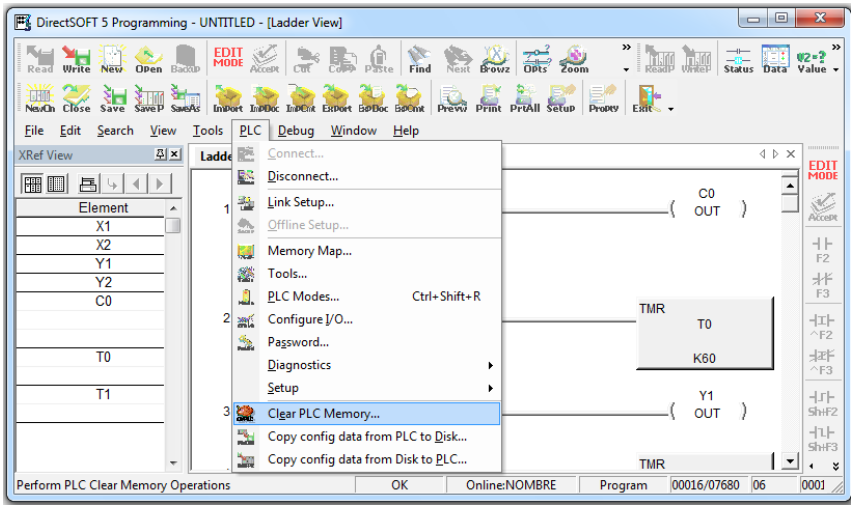
Fuente: Autores.

w) Diríjase inciso J y continúe, en caso de continuar el error siga al inciso r y seleccione otro puerto repita el proceso nuevamente.

9.1.4 Copiar un programa de DirectSOFT a la memoria del PLC. Una vez usted haya instaurado comunicación entre el PLC y el PC, y después de haber escrito el programa en *DirectSOFT*, se procede a guardar el programa en la memoria del PLC, pero Antes de hacerlo se debe limpiar la memoria del equipo, es decir borrar el programa que está guardado en el. Para borrar la memoria del PLC siga los siguientes pasos:

a) Presione clic sobre el link *PLC* en la barra de herramientas, luego en la ventana emergente que se muestra en la figura 221, presione clic sobre la opción *Clear PLC Memory.*

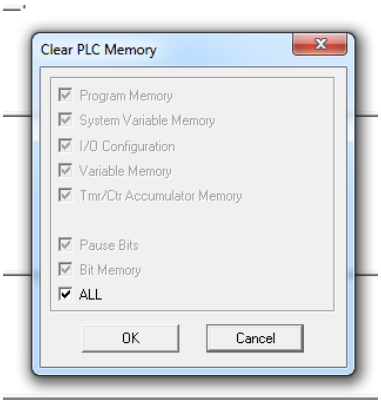
Figura 221. Ventana PLC.



Fuente: Autores.

b) En la ventana emergente que se muestra en la figura 222, seleccione *ALL*, y presione *OK*.

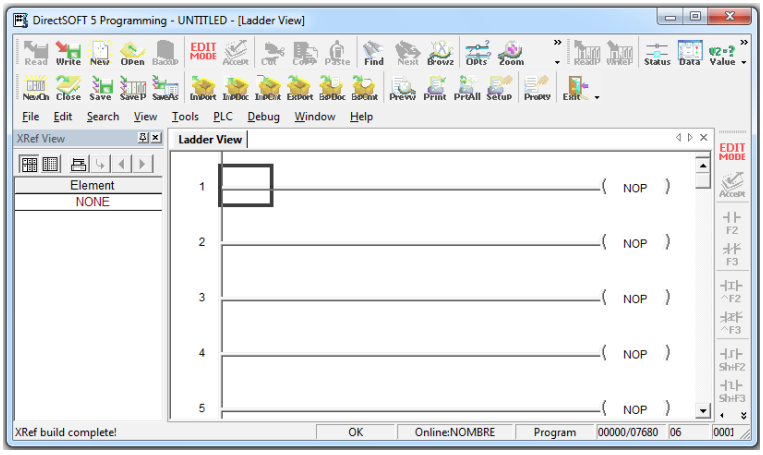
Figura 222. Ventana Clear PLC memory.



Fuente: Autores.

Ya se ha limpiado la memoria del equipo por lo tanto se verá como la figura 223. Cierre la ventana.

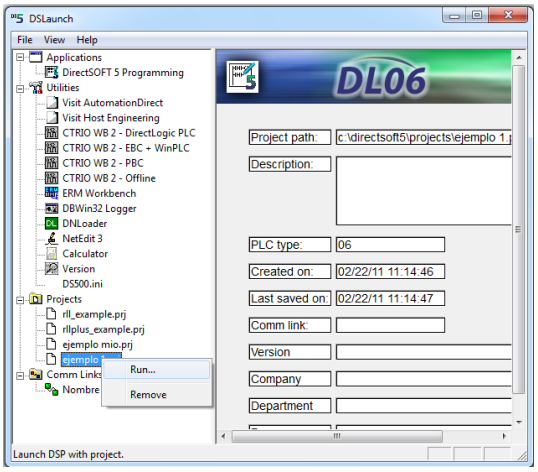
Figura 223. Memoria del PLC limpia.



Fuente: Autores.

c) Valla al programa y en la carpeta *Projects*, presione clic derecho sobre el programa que desea copiar al PLC, y pulse *Run* (ver figura 224).

Figura 224. Carpeta Project.



Fuente: Autores.

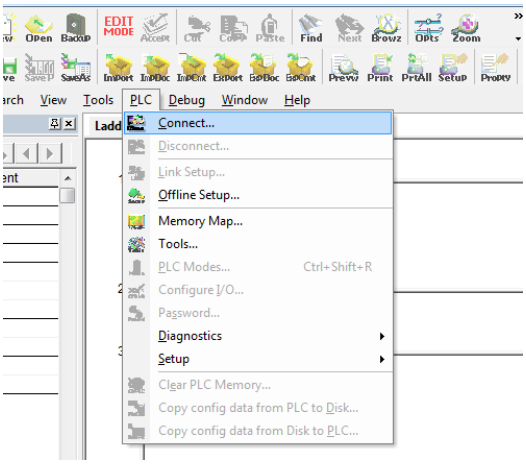
d) Presione Run DirectSOFT100 como se muestra en la figura 225.

Figura 225. Ventana de inicio para la programación.

Fuente: Autores.

e) Presione el link *PLC* en la barra de herramientas y en la ventana emergente si no está activada la opción *Diconnect* quiere decir que no hay conexión entre el software y el PLC, para establecerla presione *Connect* (ver figura 226).

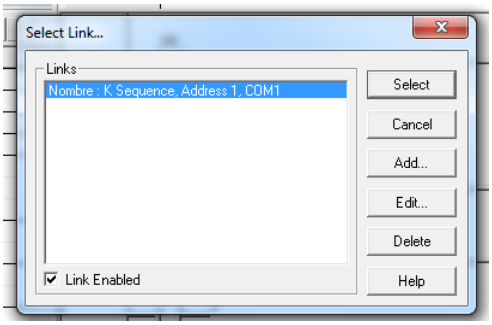
Figura 226. Ventana de conexión del PLC.



Fuente: Autores.

f) Seleccione el link de comunicación que usted creó en este caso *Nombre* y presione *select* como se muestra en la figura 227.

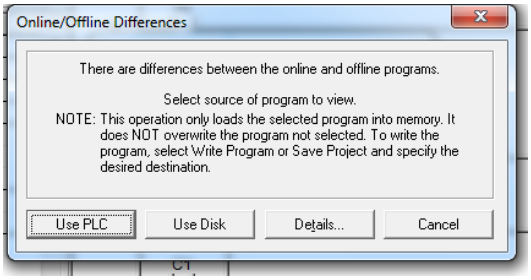
Figura 227. Ventana de selección del Link de comunicación.



Fuente: Autores.

g) En la ventana emergente que se observa en la figura 228, seleccione *Use Disk*, para abrir el programa que previamente escribió en *DirectSOFT* en este caso Ejemplo 1. Si presiona *Use PLC*, abrirá el programa que está en la memoria del PLC, en este caso no hay ninguno recuerde que previamente este se ha borrado.

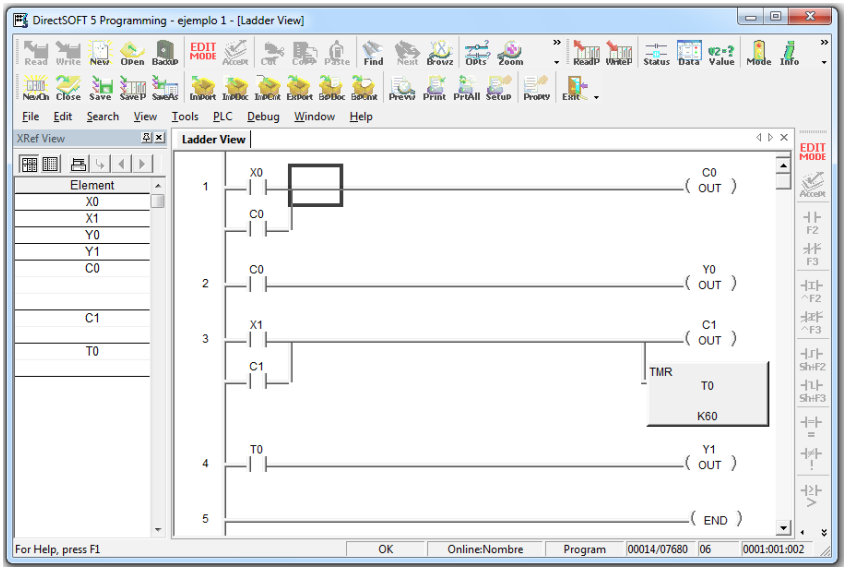
Figura 228. Ventana para la selección de la memoria.



Fuente: Autores.

Ahora puede visualizar el programa que desea guardar el PLC, en este caso el de la figura 229.

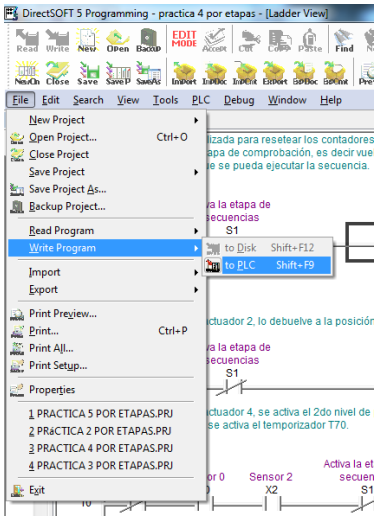
Figura 229. Programa a guardar en la memoria del PLC



Fuente: Autores.

h) Para guardar el programa en la memoria del PLC, lleve el curso hasta *File*, en la barra de herramientas, luego presione *Write Program*, en la ventana emergente que se observa en la figura 230, presione *to PLC*, y se comenzará a guardar el programa en el PLC.

Figura 230. Link para escribir el programa en le PLC.

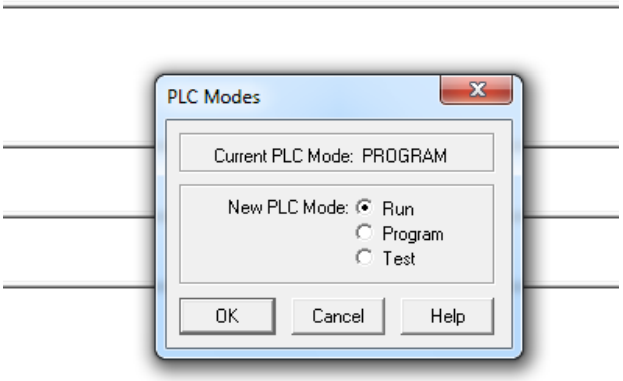


Fuente: Autores.

En caso de no aparecer activado el link *to PLC,* valla la ventana PLC verifique que aparezca activado el link *Diconnect,* de no ser así presione *Connect.* Si el link *Diconnect* aparece activado y aún así no aparece en *Write program* activado el link *to PLC,* entonces presione *Diconnect,* y luego *Connect,* para desactivar el enlace y activarlo nuevamente.

9.1.4.1 Cambiar modos del PLC mediante el programa. Esto solo se puede realizar si la conexión con el PLC está activa. Pulse sobre la ventana PLC mode situada cuarto recuadro de la parte inferior del programa, se desplegará la ventana que se observa en la figura 231, y en ella se puede seleccionar el modo del PLC.

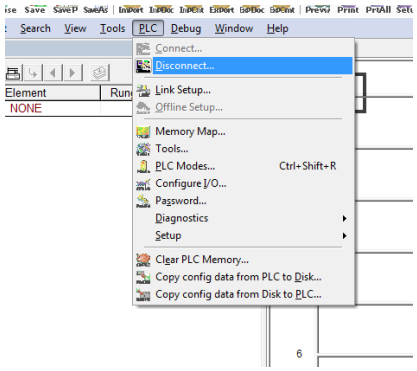
Figura 231. Ventana para la selección del modo del PLC



Fuente: Autores.

9.1.4.2 Desconectar PLC de la PC. Cada vez que desea desconectar el equipo de la computadora, debe presionar en la ventana *PLC* el link *Diconnect* que se observa en la figura 232, de esta forma se desconecta el equipo de una manera segura.

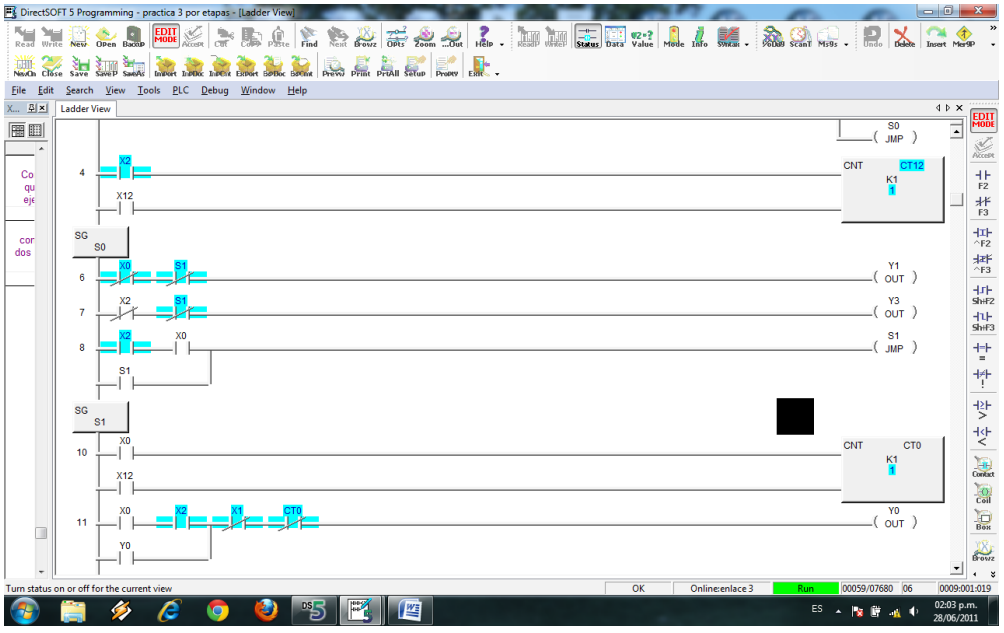
Figura 232. Link para desconectar el PLC de la PC.



Fuente: Autores.

9.1.4.3 Estado de los elementos. Se puede apreciar en tiempo real el estados de los elementos durante la ejecución de un programa como se aprecia en la figura 231, mediante la herramienta Status, para hacerlo presione el link Status, en la barra de herramienta y de inmediato los elemento que estén activados se tornaran de color azul, tal como se observa en la figura 233.

Figura 233. Visualización en tiempo real del estado de los elementos.



Fuente: Autores.

9.2 PROGRAMACIÓN DEL MICROCONTROLADOR.

Esta sección contiene una breve descripción de los principales comandos del software Arduino, su instalación y comunicación con la tarjeta de control.

9.2.1 Instalación de software de Arduino

Par instalar el software de Arduino necesitamos descargar el fichero en las siguientes direcciones:

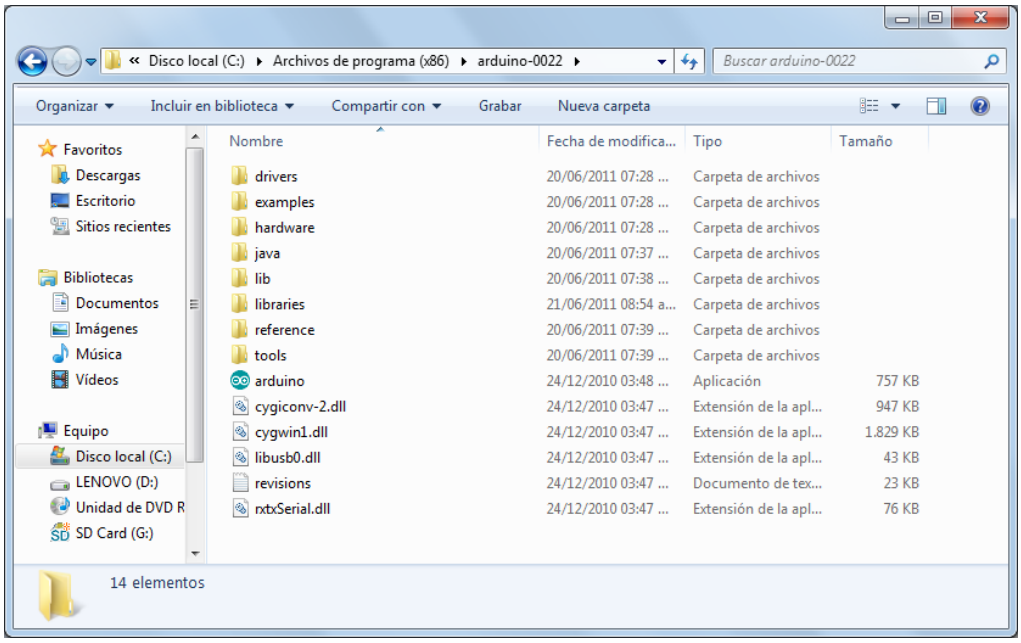
Para Windows: http://arduino.googlecode.com/files/arduino-0019.zip

Para Mac OS X: http://arduino.googlecode.com/files/arduino-0019.dmg

Para Linux: http://arduino.googlecode.com/files/arduino-0019.tgz

Luego de la descarga se descomprime el fichero en la carpeta el usuario desee, abra la carpeta que contiene los elementos que se muestran en la figura 234.

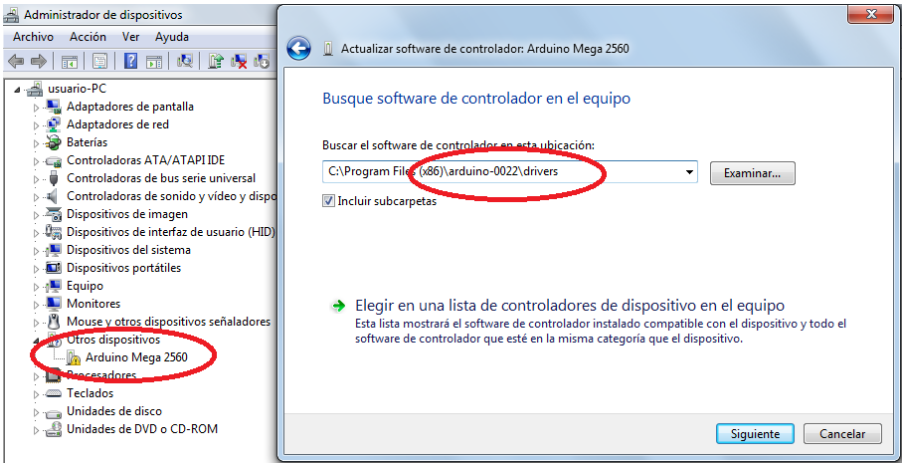
Figura 234. Elementos de la carpeta del programa.



Fuente: Autores.

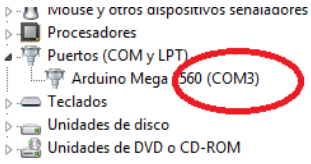
9.2.1.1 Instalación de los drivers USB. Cuando se conecta por primera vez la placa de Arduino se obtiene un cuadro de dialogo con el siguiente mensaje: "No se pudo instalar el software de controlador de dispositivo". Luego En la ventaba de administrador de dispositivos se presiona clic a la donde dice "otros dispositivos" como se observa en la figura 235, se presiona clic derecho en "Arduino Mega" y se selecciona "actualizar software de controlador", se escoge en el fichero que previamente se había descargado la carpeta "drivers", se presiona aceptar y siguiente para que proceda la instalación. Cuando termine la instalación debe aparecer el Puerto Serie (ver figura 236), por lo general COM3, COM4 o COM5.

Figura 235. Administrador de dispositivos.



Fuente: Autores.

Figura 236. Puertos series para la comunicación del Arduino con la PC.



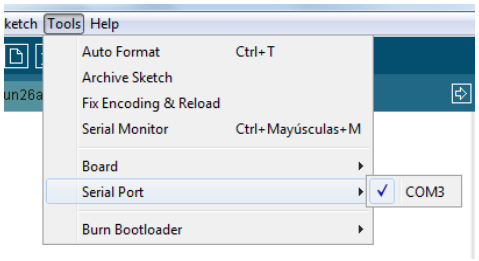
Fuente: Autores.

9.2.2 Programación de Microcontrolador. El ATmega1280 en el Arduino Mega viene precargado con un gestor de arranque (bootloader) que permite cargar nuevo código sin necesidad de un programador por hardware externo. Se comunica utilizando el protocolo STK500 original.

También te puedes saltar el gestor de arranque y programar directamente el microcontrolador a través del puerto ISCP (In Circuit Serial Programming). Si tienes un programador externo (por ejemplo un AVR-ISP STK500 o un programador paralelo puedes cargar tus sketches (programas) en la placa Arduino sin necesidad de un gestor de arranque. Esto te permite utilizar el total de la memoria disponible en el chip de la Arduino. Así en el ATmega168 dispondrías de 16 KB en lugar de los 14 KB usuales. Además te evitarás el retraso causado por el gestor de arranque cuando conectas o reseteas la placa

9.2.2.1 Entorno de la aplicación Arduino. Una vez esté instalado el software de controlador se puede ejecutar la aplicación de Arduino que se encuentra en su respectiva carpeta. Una vez este ejecutado el entorno Arduino se da click en el menú Tools > Serial Port y se selecciona el puerto serie que corresponda, para este caso (COM3), como se muestra en la figura 237.

Figura 237. Ventana Tool del software Arduino.



Fuente: Autores.

En el programa Arduino cuenta con las siguientes aplicaciones:

Barra de herramientas. En la barra de herramientas se encuentran los siguientes comandos:

 *Verify/Compile*: Chequea el código en busca de errores.

 *Stop:* Detiene los demás comandos.

 *New:* Crea una nueva rutina.

 *Open:* Muestra un menú con las rutinas guardadas.

 *Save:* Guarda las rutinas.

 *Upload to I/O board:* Carga el código a la placa Arduino.

 *Serial Monitor:* Muestra una ventana para visualización y envió de datos entre la placa y la Pc.

 *Tab Menú:* Permite gestionar rutinas en más de una pestaña.

Pestañas. En estas se gestionan las rutinas, estas pueden ser: Archivos de código de Arduino (extensión .pde), Archivos de C (extensión c), Archivos de C++ (extensión cpp,) y Archivos de cabecera (extensión .h).

Menús.

Verify/Compile: Chequea el código en busca de errores.

Show Sketch Folder: Abre la carpeta de rutinas en tu escritorio.

Import Library: Llama una librería con funciones pregrabadas que facilitan la escritura del código.

Add File... : Añade otro fichero fuente a la práctica. El nuevo archivo aparece en una nueva pestaña en la ventana de la práctica. Esto facilita y agranda los proyectos. Los archivos pueden ser eliminados de una rutina usando el Tab Menú.

Tools.

Auto Format: Esta función arregla el código para facilitar la comprensión.

Archive Sketch. Guarda el programa dentro de un archivo comprimido.

Serial Monitor: muestra una ventana para visualización y envió de datos entre la placa y el Pc.

Serial Port. Este menú contiene todos los dispositivos serie (reales o virtuales) de tu Pc. Antes de subir tu rutina, necesitas seleccionar el elemento de este menú que representa a tu placa Arduino. Para descubrirlo, busca USB serial device en la sección puertos del administrador de dispositivos de Windows.

Board. Selecciona la placa que se está usando. Esto controla la forma en que la rutina es compilada y cargada así como el comportamiento de los elementos del menú Burn Bootloader.

Burn Bootloader. Los elementos en este menú te permiten grabar un bootloader en tu placa con una variedad de programadores. Esto no es necesario si se construye una placa.

SKECHS (Programas).

Estructura. La estructura básica del lenguaje de programación Arduino se organiza en al menos dos partes o funciones que encierran bloques de declaraciones. Ambas funciones son requeridas para que el programa funcione.

```
void setup()
{
Declaraciones ;
}
void loop()
{
código;
}
setup()
```

La función setup debe contener la declaración de cualquier variable al comienzo del programa. Es la primera función a ejecutar en el programa, es ejecutada una vez y es usada para asignar pinMode o inicializar las comunicaciones serie.

```
void setup()
{
pinMode(pin, OUTPUT); //ajusta 'pin' como salida
}
loop()
```

La función loop se ejecuta a continuación e incluye el código que se ejecuta continuamente leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas Arduino y hace la mayor parte del trabajo.

```
void loop()
{
digitalWrite(pin, HIGH); //Activa 'pin'
delay(1000); //espera un segundo
digitalWrite(pin, LOW); //Desactiva 'pin'
delay(1000); //espera un segundo
}
```

Funciones. Una función es un bloque de código que tiene un nombre y un grupo de declaraciones que se ejecutan cuando se llama a la función. Se pueden hacer uso de funciones integradas como void setup() y void loop() o escribir nuevas. Las funciones se escriben para ejecutar tareas repetitivas y reducir el desorden en un programa. En primer lugar se declara el tipo de la función, que será el valor retornado por la función (int, void...). A continuación del tipo, se declara el nombre de la función y, entre paréntesis, los parámetros que se pasan a la función. La siguiente función int delayVal(), asigna un valor de retardo en un programa por lectura del valor de un potenciómetro.

```
int delayVal()
{
int v; //crea una variable temporal 'v'
v = analogRead(pot); //lee el valor del potenciómetro
v /= 4; //convierte 0-1023 a 0-255
return v; //devuelve el valor final de v
}
```

llaves {}. Las llaves definen el comienzo y el final de bloques de función y bloques de declaraciones como void loop() y sentencias for e if. Las llaves deben estar balanceadas (a una llave de apertura {debe seguirle una llave de cierre }). Las llaves no balanceadas provocan errores de compilación. El entorno Arduino incluye una característica para chequear el balance de llaves. Sólo selecciona una llave y su compañera lógica aparecerá resaltada.

punto y coma (;). Un punto y coma debe usarse al final de cada declaración y separa los elementos del programa. También se usa para separar los elementos en un bucle for. Olvidar un punto y coma al final de una declaración producirá un error de compilación.

```
int x = 13;    //declara la variable 'x' como el entero 13
```

bloques de comentarios /...*/.* Los bloques de comentarios, o comentarios multilínea, son áreas de texto ignoradas por el programa y se usan para grandes descripciones de código o comentarios que ayudan a otras personas a entender partes del programa. Empiezan con /* y terminan con */ y pueden abarcar múltiples líneas.

```
/*
este es un bloque de comentario encerrado
no olvides cerrar el comentario
tienen que estar balanceados!
*/
```

Como los comentarios son ignorados por el programa y no ocupan espacio en memoria deben
usarse generosamente y también pueden usarse para _comentar_ bloques de código con propósitos de depuración.

Comentarios de línea (//). Comentarios de una línea empiezan con // y terminan con la siguiente línea de código. Como el bloque de comentarios, son ignorados por el programa y no toman espacio en memoria.

Comentarios de una línea se usan a menudo después de declaraciones válidas para proporcionar más información sobre qué lleva la declaración o proporcionar un recordatorio en el futuro.

```
// este es un comentario de una línea
```

Variables. Una variable es una forma de llamar y almacenar un valor numérico para usarse después por el programa. Como su nombre indica, las variables son números que pueden cambiarse continuamente al contrario que las constantes,

cuyo valor nunca cambia. Una variable necesita ser declarada y, opcionalmente, asignada al valor que necesita para ser almacenada.

```
int inputVariable = 0; //declara una variable y asigna el valor a 0
inputVariable = analogRead(2); //ajusta la variable al valor del pin analógico 2
```

Una vez que una variable ha sido asignada, o reasignada, puedes comprobar su valor para ver si cumple ciertas condiciones, o puedes usarlo directamente.

```
if(inputVariable < 100) //comprueba si la variable es menor que 100
{
inputVariable = 100; //si es cierto asigna el valor 100
}
delay(inputVariable); //usa la variable como retardo
```

Declaración de variable. Todas las variables tienen que ser declaradas antes de que puedan ser usadas. Declarar una variable significa definir su tipo de valor, como int, long, float, etc., definir un nombre específico, y, opcionalmente, asignar un valor inicial. Esto sólo necesita hacerse una vez en un programa pero el valor puede cambiarse en cualquier momento usando aritmética y varias asignaciones.

```
int inputVariable = 0;
```

Una variable puede ser declarada al comienzo del programa antes del void setup() o localmente dentro de funciones. Donde la variable es declarada determina el espacio de la variable.

Una variable global es una que puede ser vista y usada por cualquier función y declaración en un programa. Esta variable se declara al comienzo del programa, antes de la función setup().

Una variable local es la que se define dentro de una función o como parte de un bucle for. Sólo es visible y sólo puede ser usada dentro de la función en la cual fue declarada. Además, es posible tener dos o más variables del mismo nombre en diferentes partes del programa que contienen diferentes valores.

```
int value; //'value' es visible por cualquier función
void setup()
{
//no se necesita setup
}
void loop()
{
for(int i=0; i<20;) //'i' es sólo visible dentro del bucle for
{
i++;
}
float f; //'f' es sólo visible dentro de loop
}
```

Tipos de datos:

Byte. Byte almacena un valor numérico de 8 bits sin puntos decimales. Tienen un rango de 0 a 255.

```
byte someVariable = 180; //declara 'someVariable' como un tipo byte
```

int. Enteros son los tipos de datos primarios para almacenamiento de números sin puntos decimales y almacenan un valor de 16 bits con un rango de -32,768 a 32,767.

```
int someVariable = 1500; //declara 'someVariable' como tipo int
```

long. Tipo de datos de tamaño extendido para enteros largos, sin puntos decimales, almacenados en un valor de 32 bits con un rango de -2,146,483,648 a 2,147,483,647.

```
long someVariable = 90000; //declara 'someVariable' como tipo long
```

float. Un tipo de datos para números en punto flotante, o números que tienen un punto decimal. Los números en punto flotante tienen mayor resolución que los enteros y se almacenan como valor de 32 bits con un rango de -3.4028235E+38 a 3.4028235E+38.

```
float someVariable = 3.14; //declara 'someVariable' como tipo float
```

arrays. Un array es una colección de valores que son accedidos con un índice numérico. Cualquier valor en el array debe llamarse escribiendo el nombre del array y el índice numérico del valor. Los arrays están indexados a cero, con el primer valor en el array comenzando con el índice número 0. Un array necesita ser declarado y opcionalmente asignarle valores antes de que puedan ser usados.

```
int myArray[] = {value0, value1, value2...};
```

Asimismo es posible declarar un array declarando el tipo del array y el tamaño y luego asignarle valores a una posición del índice.

```
int myArray[5]; //declara un array de enteros con 6 posiciones
myArray[3] = 10; //asigna a la cuarta posición del índice el valor 10
```

Para recibir un valor desde un array, asignamos una variable al array y la posición del índice:

```
x = myArray[3]; //x ahora es igual a 10
```

Aritmética. Los operadores aritméticos incluyen suma, resta, multiplicación y división. Retornan la suma, diferencia, producto o cociente (respectivamente) de dos operandos.

```
y = y+3;
x = x-7;
i = j*6;
r = r/5;
```

La operación es llevada a cabo usando del tipo de datos de los operandos, así 9/4 devuelve 2 en lugar de 2.25. Si los operandos son de tipos diferentes, el tipo mayor es usado para el cálculo.

Asignaciones compuestas. Las asignaciones compuestas combinan una operación aritmética con una asignación de variable.
Estas son muy frecuentemente encontradas en bucles for. Las asignaciones compuestas más comunes incluyen:

```
x++; //lo mismo que x = x+1
x--; //lo mismo que x = x-1
x += y; //lo mismo que x = x+y
x -= y; //lo mismo que x = x-y
x *= y; //lo mismo que x = x*y
x /= y; //lo mismo que x = x/y
```

Operadores de comparación. Las comparaciones de una variable o constante con otra se usan a menudo en declaraciones if para comprobar si un condición específica es cierta.

```
x == y; //x es igual a y
x != y; //x no es igual a y
x < y; //x es menor que y
x > y; //x es mayor que y
x <= y; //x es menor o igual que y
x >= y; //x es mayor o igual que y
```

Operadores lógicos. Los operadores lógicos son normalmente una forma de comparar dos expresiones y devuelven TRUE o FALSE dependiendo del

operador. Hay tres operadores lógicos, AND, OR y NOT, que se usan a menudo en declaraciones if.

```
//AND lógico:
if(x>0 && x<5) //verdadero sólo si las dos expresiones son ciertas
//OR lógico:
if(x>0 || y>0) //verdadero si al menos una expresión es cierta
//NOT lógico:
if(!(x>0)) //verdadero sólo si la expresión es falsa
```

Constantes. El lenguaje Arduino tiene unos cuantos valores predefinidos que se llaman constantes. Se usan para hacer los programas más legibles. Las constantes se clasifican en grupos.

true/false. Estas son constantes Booleanas que definen niveles lógicos. FALSE se define como 0 (cero) mientras TRUE es 1 o un valor distinto de 0.

```
if(b == TRUE)
{
Hacer algo;
}
```

high/low. Estas constantes definen los niveles de pin como HIGH o LOW y se usan cuando se leen o se escriben los pines digitales. HIGH está definido como el nivel 1 lógico, ON ó 5 V, mientras que LOW es el nivel lógico 0, OFF ó 0 V.

```
digitalWrite(13, HIGH);
```

input/output. Constantes usadas con la función pinMode() para definir el modo de un pin digital como INPUT u OUTPUT.

```
pinMode(13, OUTPUT);
```

Control de flujo

if. Las sentencias if comprueban si cierta condición ha sido alcanzada y ejecutan todas las sentencias dentro de las llaves si la declaración es cierta. Si es falsa el programa ignora la sentencia. Hay que tener cuidado en usar (=) en lugar de (==)

```
if(someVariable ?? value)
{
hacer algo;
}
```

```
If/else.  Permite tomar decisiones
```

```
if(inputPin == HIGH)
{
Hacer A;
}
else
{
Hacer B;
}
```

Else. Puede preceder a otra comprobación if, por lo que múltiples y mutuas comprobaciones exclusivas pueden ejecutarse al mismo tiempo.

```
if(inputPin < 500)
{
Hacer A;
}
else if(inputPin >= 1000)
{
Hacer B;
}
else
{
Hacer C;
}
```

for. La sentencia for se usa para repetir un bloque de declaraciones encerradas en llaves un número específico de veces. Un contador de incremento se usa a menudo para incrementar y terminar el bucle. Hay tres partes separadas por punto y coma (;), en la cabecera del bucle.

```
for(inicialización; condición; expresión)
{
hacer algo;
}
```

La inicialización de una variable local, o contador de incremento, sucede primero y una sola una vez. Cada vez que pasa el bucle, la condición siguiente es comprobada. Si la condición devuelve TRUE, las declaraciones y expresiones que siguen se ejecutan y la condición se comprueba de nuevo. Cuando la condición se vuelve FALSE, el bucle termina.

```
for(int i=0; i<20; i++) //declara i, comprueba si es menor
{ //que 20, incrementa i en 1
digitalWrite(13, HIGH); //activa el pin 13
delay(250); //pausa por un 1/4 de segundo
digitalWrite(13, LOW); //desactiva el pin 13
delay(250); //pausa por un 1/4 de segundo
}
```

while. El bucle while se repetirá continuamente, e infinitamente, hasta que la expresión dentro del paréntesis se vuelva falsa. Algo debe cambiar la variable testeada, o el bucle while nunca saldrá.
Esto podría estar en tu código, como por ejemplo una variable incrementada, o una condición externa, como un sensor de comprobación.

```
while(someVariable ?? value)
{
hacer algo;
}
```

```
while(someVariable < 200) //comprueba si es menor que 200
{
hacer algo; //ejecuta las sentencias encerradas
someVariable++; //incrementa la variable en 1
}
```

do/ while. El bucle do/ while es un bucle que trabaja de la misma forma que el bucle while, con la excepción de que la condición es testeada al final del bucle, por lo que el bucle do/ while siempre
se ejecutará al menos una vez.

```
do
{
hacer algo;
}while(someVariable ?? value);
do
{
x = readSensors(); //asigna el valor de readSensors() a x
delay(50); //pausa de 50 milisegundos
}while(x < 100); //repite si x es menor que 100
```

E/S digitales.

pinMode(pin, mode)**.** Se usa en void setup() para configurar un pin específico para que se comporte o como INPUT o como OUTPUT.

```
pintMode(pin, OUTPUT); //ajusta 'pin' como salida
```

Los pines digitales de Arduino están ajustados a INPUT por defecto, por lo que no necesitan ser declarados explícitamente como entradas.

pinMode(). Los pines configurados como INPUT se dice que están en un estado de alta impedancia.

Hay también convenientes resistencias de pull-up de 20KOhm, integradas en el chip ATmega que pueden ser accedidas por software. A estas resistencias pull-up integradas se accede de la siguiente manera.

```
pinMode(pin, INPUT); //ajusta 'pin' como entrada
digitalWrite(pin, HIGH); //activa la resistencia de pull-up
```

Las resistencias de pull-up se usarían normalmente para conectar entradas como interruptores. Los pines configurados como OUTPUT se dice que están en un estado de baja impedancia y pueden proporcionar 40 mA a otros dispositivos/circuitos.

digitalRead(pin). Lee el valor desde un pin digital especificado con el resultado HIGH o LOW. El pin puede ser especificado o como una variable o como una constante (0 - 13).

```
value = digitalRead(Pin); //ajusta 'value' igual al pin de entrada
```

digitalWrite(pin, value). Devuelve o el nivel lógico HIGH o LOW a (activa o desactiva) un pin digital especificado. El pin puede ser especificado como una variable o constante (0 - 13).

```
digitalWrite(pin, HIGH); //ajusta 'pin' a HIGH
```

E/S analógicas.

analogRead(pin). Lee el valor desde un pin analógico especificado con una resolución de 10 bits. Esta función sólo trabaja en los pines analógicos (0 - 5). Los valores enteros devueltos están en el rango de 0 a 1023.

```
value = analogRead(pin); //ajusta 'value' igual a 'pin'
```

analogWrite(pin, value). Escribe un valor pseudo analógico usando modulación por ancho de pulso (PWM en inglés) a un pin de salida marcado como PWM. El valor puede ser especificado como una variable o constante con un valor de 0 a 255.

```
analogWrite(pin, value); //escribe 'value' al 'pin' analógico
```

El valor de salida varía de 0 a 5 V según el valor de entrada (de 0 a 255) en función del tiempo de pulso. Si t es el tiempo de pulso, la tabla 5.1 muestra la equivalencia entre el valor y la salida en función del tiempo.
Como esta es una función hardware, el pin generará una onda estática después de una llamada a analogWrite en segundo plano hasta la siguiente llamada a analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin).

```
int led = 10; //LED con una resistencia de 220ohm en el pin 10
int pin = 0; //potenciómetro en el pin analógico 0
int value; //valor para lectura
void setup(){} //setup no es necesario
void loop()
{
value = analogRead(pin); //ajusta 'value' igual a 'pin'
value /= 4; //convierte 0-1023 a 0-255
analogWrite(led, value); //saca la señal PWM al led
}
```

Tiempo.

delay(ms). Pausa tu programa por la cantidad de tiempo especificada en milisegundos, donde 1000 es igual a 1 segundo.

```
delay(1000); //espera por un segundo
```

millis(). Devuelve el número de milisegundos desde que la placa Arduino empezó a ejecutar el programa actual como un valor long sin signo.

```
value = millis(); //ajusta 'value' igual a millis()
```

Nota: Este número se desbordará (resetear de nuevo a cero), después de aproximadamente 9 horas.

Matemáticas.

min(y). Calcula el mínimo de dos números de cualquier tipo de datos y devuelve el número más pequeño.

```
value = min(value, 100); //asigna a 'value' al más pequeño de 'value' o 100,
asegurándose que nunca superara 100.
```

max(y). Calcula el máximo de dos números de cualquier tipo de datos y devuelve el número más grande.

```
value = max(value, 100); //asigna a 'value' al más grande de 'value' o 100,
asegurándose que es al menos 100.
```

10. DETECCIÓN DE FALLAS

10.1 PRINCIPALES CAUSAS DE FALLAS

Pueden existir muchas causas que provoque falla, entre las más comunes tenemos:

- Problemas de Operario.
- Fallas en el suministro de potencia.
- Problemas mecánicos.
- Problemas debidos a Ruidos.

10.1.1Problemas de Operario. Estos ocurren debido al uso incorrecto por parte de la persona que utiliza el equipo. Uno de los motivos es la falta de conocimiento adecuado del funcionamiento del equipo, lo que en ocasiones conlleva a suponer que el equipo opera incorrectamente, cuando en realidad no existen problemas de funcionamiento como tal. Tales situaciones ocurren frecuentemente por lo que deben ser una de las primeras instancia que se verifiquen.

10.1.2 Fallas en el suministro de potencia. La falta de suministro de potencia en los componentes significa el cese de funcionamiento de los mismos, esto puede ocurrir por múltiples factores que van desde un el corto de suministro en la red principal, hasta fallas en los componentes. En caso de ocurrir este tipo de fallas se deben inspeccionar las líneas en orden jerárquico.

10.1.3 Problemas mecánicos. Son todos aquellos que surgen debido a desperfectos en componentes de tipo mecánico tales como: conectores, relevos, pulsadores, sensores, etcétera. En caso de ocurrir este tipo de fallas se debe examinar con detenimiento los elementos que se consideran están fallando.

10.1.4 Problemas debidos a Ruidos. El ruido esto toda señal externa que dentro del equipo puede ser causal de operación incorrecta, este es una fuente potencial importante de problemas en los en los circuitos digitales. Las señales de ruido pueden provenir de transitorios en las líneas de corriente alterna o de campo magnético o eléctrico originados en equipos aledaños, así como de interferencias debidas a transmisiones de radio o de televisión. Para evitar este tipo de problemas, se debe comprobar que el ambiente de funcionamiento de los componentes está libre de los mismos, en caso de su presencia se deben tomar las medidas pertinentes para cada caso.

10.2 PROCEDIMIENTOS PARA LA SOLUCIÓN DE PROBLEMAS[13]

La reparación de equipos electrónicos puede resumirse cuatro (4) sencillos pasos:

1. Recolección de Datos.
2. Localizar el problema.
3. Efectuar la reparación.
4. Verificación de la operación correcta.

10.2.1Recolección de Datos: Es aquella en la cual se hace acopio de toda la información pertinente al equipo bajo observación. Por ejemplo, lo primero que debe hacerse es obtener la documentación, en la cual se incluye tanto los diagramas esquemáticos circuitales así como los manuales de servicio, información de calibración y similares.

10.2.2 Localizar el problema. Es por lo general es lo más difícil, el grado de dificultad y la cantidad de tiempo que esta fase del problema consuma, dependen de la complejidad del equipo y la naturaleza del daño. Los siguientes pasos pueden ayudar a desarrollar un método sistemático para localizar la avería:

[13] Fuente: http://www.monografias.com/trabajos19/diagnostico-de-fallas/diagnostico-de-fallas.shtml

a. Verifique lo obvio y sencillo primero que todo, como fusible, tomas, interruptores, etc.
b. Corra los programas de diagnostico si los hay.
c. Utilice sus sentidos, mirando, oliendo y tocando en busca de temperaturas anormales, elementos quemados, etc.
d. Verifique que los niveles de AC y DC sean correctos.
e. Utilice métodos de rastreo de señal.
f. Ensaye sustituciones sencillas de componentes o de tarjetas en cuanto sea posible.

10.2.3 Efectuar la reparación. Una vez localizado el problema se procede a la reparación, esta debe ser realizada por personal capacitado.

10.2.4 Verificación de la operación correcta. Terminada la reparación se debe verificar el óptimo funcionamiento de sistema.

10.3 DETECCIÓN Y SOLUCIÓN DE LAS FALLAS MÁS FRECUENTES

En la tabla 28 se exponen las posibles fallas que puede presentar el sistema, así como también las soluciones más inmediatas para las mismas.

Tabla 28. Identificación de riesgos operacionales (HAZOP).

Falla	Causa	Solución
No enciende el sistema	• No está accionado el selector de “ENCENDER”. • No está conectado el banco. • No están en estado ON los breakers del laboratorio. • No está en estado ON el breakers interno del banco.	• Accione el selector de “ENCENDER”. • Conectar el banco a la toma de corriente • Ubique en estado ON los breakers del laboratorio. • Abra el tablero eléctrico del banco y ubique en estado ON el breakers principal.
No enciende el PLC.	• No está accionado el selector hacia en modo “PLC”. • No está conectada la toma Principal del PLC. • Está desactivado el breakers para el PLC.	• Accione el selector al modo “PLC”. • Conecte la toma principal del PLC. • Abra el tablero eléctrico y ubique en estado ON el breakers del PLC.
No enciende el Microcontrolador.	• No está accionado el selector hacia en modo Microcontrolador • No está conectada la toma de energía del Microntronlador. • Fuente de tensión externa del Microcontrolador desconectada.	• Accione el selector al modo “Microcontrolador”. • Conecte la toma de energía del Microcontrolador. • Abra el tablero de potencia y verifique que la fuente de tensión de 12 VDC funciona correctamente.
Motor no enciende.	• Error en la programación. • No están activados los relevos auxiliares.	• Verifique la programación. • Verifique que está conectado el cable Auxiliar • Abra el tablero eléctrico y verifique que la fuente de 12 VDC esté funcionando correctamente.

Fuente: Autores.

No hay señal de los sensores	• No está conectado el cable de señales auxiliares. • No está conectado el cable de señales de entada. • Verifique su estado.	• Conecte el cable de señales auxiliares. • Conecte el cable de señales de entrada. • Para verificar el estado del sensor siga las siguientes instrucciones. a) Verifique si el indicado luminoso ubicado en el sensor se enciende al detectar un objeto. b) Verifique si el indicador luminoso correspondiente a ese sensor en las entradas del PLC está encendido, De no ser así, verifique las conexiones de los bornes. c) Verifique si el indicador luminoso correspondiente a ese sensor en la tarjeta de interfaz de señales de entrada está encendido, De no ser así, verifique las conexiones de los bornes.
No se accionan las electroválvulas.	• Cable de señales de salida desconectado. • Error en la programación. • No hay alimentación en la tarjeta de potencia. • Verificar estado de las señales.	• Conecte el cable de señales de salida. • Verifique la programación. • Verifique que las tenciones requeridas para el funcionamiento de la tarjeta de potencia estén conectadas. • Para verificar el estado de las señales siga las siguientes instrucciones. d) Corra el programa y Verifique si el indicador luminoso correspondiente en la tarjeta de potencia está encendido, si lo está, verifique las conexiones de los bornes. e) Para el PLC, corra el programa y verifique que el indicador luminoso de las

		salidas del PLC está encendido.
No se mueven los actuadores.	• No hay presión en el sistema. • No se accionan las electroválvulas.	• Verifique que se halla programado la válvula de presión. • Verificar en esta tabla el error “No se accionan las electroválvulas”.

10.4 CÓDIGOS DE ERROR EN LA PROGRAMACIÓN DEL PLC DL06

La tabla 29 muestra una lista códigos del de errores que se muestran en la pantalla del programa durante la transición del modo Program a RUN.

Tabla 29. Códigos de error en la programación del PLC.

Código	Descripción
E4**	No hay un programa en la CPU
E401	Falta una instrucción END
E402	Falta un LBL
E403	Falta un RET
E404	Falta un FOR
E405	Falta un NEXT
E406	Falta un IRT
E412	SBR / LBL >64
E421	Referencia de etapas duplicada
E422	Referencia deSBR/LBL duplicada
E423	Existe un lazo NEXT/LOOP en otro
E431	Dirección inválida ISG/SG
E433	Dirección inválida ISG / SG
E434	Dirección inválida RTC
E435	Dirección inválida RT
E436	Dirección inválida INT
E437	Dirección inválida IRTC

Código	Descripción
E438	Dirección inválida IRT
E440	IDirección inválida de datos
E441	ACON/NCON en elcuerpo principal del programa
E451	Numeración incorrecta de MLS/MLR
E453	Falta un temporizador o contador
E454	Uno de los contactos de TMRA está faltando
E455	Uno de los contactos de CNT está faltando
E456	Uno de los contactos de SR está faltando
E461	Mas de 9 niveles han sido almacenados en el stack
E462	No hay un almacenamiento correecto en el stack
E463	No se ha usado una instrución STR/STRN en renglón
E464	Falta un circuito en el programa
E471	Referencia de bobina duplicada
E472	Referencia de temporizador duplicada
E473	Referencia de contador duplicada
E499	Uso de la instruccion PRINT inválida

Fuente: Automation Direct, Manual de instrucciones KOYO DL06 Micro PLC; 2da edición, 2009. p. 305.

11.CONCLUCIONES

- Se diseño y construyó un banco de experimentación que implementa un Controlador Lógico Programable KOYO DL06 DD1 y una tarjeta de control Arduino Mega la cual tiene un Microcontrolador ATMEL ATMEGA 1280 para el control secuencial del sistema electrohidráulico del banco de prensas y malacate del laboratorio de potencia fluida.

- El sistema de control se adaptó de una manera segura y paralela al sistema de control existente, el cual se basa en lógica cableada por medio de contactores de potencia, esto permite que los estudiantes de potencia fluida e ingeniería de control realicen experiencias utilizando cualquiera de los dos sistemas de control, identificando las diferencias entre ellas y actualizando sus conocimientos respecto a las alternativas de control utilizadas actualmente en la industria.

- Se elaboró un soporte documental que consiste en un manual de prácticas, un manual de programación y un manual teórico, cuyo diseño estructural brinda las herramientas necesarias para que se aproveche al máximo el potencial que el banco didáctico puede ofrecer.

- Por medio de la elaboración de este trabajo de grado se realizó un aporte significativo a la escuela de ingeniería mecánica de la Universidad Industrial de Santander, implementando dispositivos ampliamente utilizados en la industria, lo cual se constituye en herramientas didácticas al servicio de los estudiantes, en pro de mejorar sus conocimientos y habilidades en el área de sistemas de control basados automatismos programables.

- Se logó un diseño que superó significativamente aspectos económicos y académicos de cualquier propuesta de solución comercial, demostrando que

la ejecución de este tipo de proyectos aportan de una manera relevante en la formación profesional de los estudiantes.

- Debido a que el circuito eléctrico que hace parte del banco de prensas y malacate no se encuentra en buen estado, implementar el nuevo sistema de control en forma paralela al existente, implicó un estudio riguroso de las alternativas de conexiones eléctricas diseñadas para tal fin, de tal forma que la adaptación se realizara de una manera segura, al mismo tiempo que la funcionalidad de cualquiera de los dos sistemas de control no se viera afectada.

12.RECOMENDACIONES

- Se recomienda reparar el actuador tres, ya que este presenta fugas de aceite hidráulico a través de sus sellos lo que impide su optimo funcionamiento y por ende la normal ejecución de la práctica cinco.

- Se recomienda instalar en el PLC un modulo de comunicación de Ethernet H0-ECOM100 para que la comunicación con el programador (PC) se realice de una manera más rápida, adicionalmente este modulo permitiría en un futuro la implementación de un sistema SCADA.

- Se recomienda instalar en la tarjeta de control Arduino Mega una tarjeta de comunicación Arduino Ethernet para que la comunicación con el programador (PC) se realice de una manera más rápida, adicionalmente esta tarjeta permitiría en un futuro la implementación de un sistema SCADA.

BIBLIOGRAFÍA

Automationdirect, Manual de instrucciones PLC DL06, Volumen 1 y 2, Segunda edición.
http://www.automationdirect.com/static/manuals/d006userm/d006userm.html

Balcells, josep: Romeral, José Luis. “Autómatas programables”, Alfaomega Marcombo Boixareu Editores, 2000.

Benjamin C. Kuo**.** Sistemas de Control Automáticos, Prentice Hispanoamerica Sa 1996.

Borrás P, C., Sánchez, M., Sutton, W.H., “Fluid Power System Design for Super-Gas^{TM} Fueling Stations”, 5th Japan Fluid Power System International symposium, Nov. 13-16, 2002, Nara, Japan.

Borrás, C., Sánchez, M., Sutton, W.H., “Automatic Control System for Composite Fuel Station”, 2002-01-1653 SAE International “Spring” Fuels & Lubricants. May 6. Reno Nevada 2002. USA.

Ludy Jimena Diaz Suarez, Jose Nolberto Rincon Rodriguez, Accionamiento de motores eléctricos basado en lógica programada, Diseño y construcción de bancos didácticos, Publicaciones UIS 2010.

Meneses Flórez, Jorge Enrique. “Autómatas programables Industriales”, Universidad Industrial de Santander.

Meneses Flórez, Jorge Enrique. “Documentación asignatura autómatas programábles”, Universidad Indistrila de Santander.

Sánchez, M., Borrás, C., Hergenrether, D., Sutton, W.H., “Super Gas™ Vehicle Conversion”, SAE FTT, Paper Number: 2001-01-2473,August 20, 2001.

Sánchez, M.., Borrás, C., Hergenrether, D., Sutton, W.H., Chandra H., “Super Gas™ Fueling Stations”, SAE FTT, Paper Number: 2001-01-2472, August 20, 2001.

Saúl Fernando Flores Gomez, Diseño, elaboración e implementación de prácticas para el laboratorio de sistemas oleoneumáticos (potencia fluida) adscrito a la escuela de Ingeniería mecánica, Publicaciones UIS 2004

Schneider Electric, Manual electrotécnico Telesquemario, Tecnologías de control industrial. Junio 1999. 289p.

.

ANEXOS

ANEXO A. ESPECIFICACIONES TECNICAS DEL PLC KOYO DL06 DD1

Especificaciones generales del PLC DL06 DD1

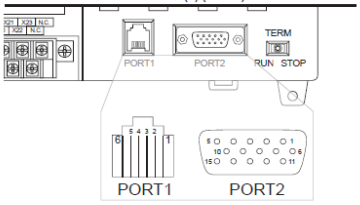
Especificaciones generales D0-06DD1	
Requerimientos de alimentación	100 – 240 VCA, 40 VA máximo,
Puerto de comunicación 1, 9600 baud (Fijo), 8 bits de datos, 1 bit stop, paridad Odd	K–Sequence (Esclavo), DirectNET (Esclavo), MODBUS (Esclavo)
Puerto de comunicación 2, 9600 baud (original),, 8 bits de datos, 1 bit stop, paridad Odd	K–Sequence (Esclavo), DirectNET (Maestro/Esclavo), MODBUS (Maestro/Esclavo), Non-sequence / print, ASCII
Tipo de cable de programación	D2–DSCBL
Temperatura de operación	32 a 131° F (0 a 55 C)
Temperatura de almacenamiento	–4 a 158° F (–20 a 70 C)
Humedad relativa	5 a 95% (sin condensación)
Calidad del aire ambiente	No se permite en ambientes con gases corrosivos
Vibración	MIL STD 810C 514.2
Choque	MIL STD 810C 516.2
Inmunidad al ruido	NEMA ICS3–304
Tipo de terminal	Removible
Sección del cable	Un AWG#16 o dos AWG#18, AWG#24 mínimo

Especificaciones de entradas CC		
Parámetro	Entradas HSIO, X0 – X3	Entradas normales CC X4 – X23
Min. - Max. Voltage Range	10,8 – 26,4 VCC	10,8 – 26,4 VCC
Voltaje de operación Range	12 – 24 VCC	12 – 24 VCC
Voltaje de cresta	30 VCC (máxima frecuencia 7 kHz)	30 VCC
Ancho de pulso mínimo	100 µs	N/A
Voltaje de detección de estado ON	> 10,0 VCC	> 10,0 VCC
Voltaje de estado OFF	< 2,0 VCC	< 2,0 VCC
Max. Corriente en las entradas	6 mA @12VCC, 13mA @24VCC	4 mA @12VCC, 8.5mA @24VCC
Impedancia de entradas	1.8 Ωk @ 12 – 24 VCC	2.8 Ωk @ 12 – 24 VCC
Corriente mínima en estado ON	>5 mA	>4 mA
Corriente máxima en estado OFF	< 0,5 mA	<0,5 mA
Respuesta cuando va de OFF a ON	<70 µs	2 – 8 ms, 4 ms típico
Respuesta cuando va de ON a OFF	<70 µs	2 – 8 ms, 4 ms típico
Indicadores LED de estado	Lado lógico	Lado lógico
Comunes	4 canales / común x 5 bancos no aislados	

Especificaciones de salidas CC		
Parámetro	Salidas de pulsos Y0 – Y1	Salidas normales Y2 – Y17
Voltaje mínimo y máximo	5 – 30 VCC	5 – 30 VCC
Voltaje de operación	6 – 27 VCC	6 – 27 VCC
Voltaje de cresta	< 50 VCC (10 kHz max. frequency)	< 50 VCC
Caída de tensión cuando es ON	0,3 VCC @ 1 A	0,3 VCC @ 1 A
Corriente máxima(resistiva)	0,5 A/pt., 1A / pto.como salida normal.	1,0 A / punto
Máxima corriente de fuga	15 µA @ 30 VCC	15 µA @ 30 VCC
Máxima corriente de inrush	2 A por 100 ms	2 A por 100 ms
Alimentación externa CC	20 - 28 VCC; 150mA máximo	20 - 28 VCC Max 280 mA (Aux. 24VCC alimenta terminal V+
Respuesta cuando va de OFF a ON	< 10µ s	< 10 µs
Respuesta cuando va de ON a OFF	< 20 µs	< 60 µs
Indicadores LED de estado	Lado lógico	Lado lógico
Comunes	4 canales / común x 4 bancos no aislados	
Fusibles	Ninguno (se recomienda colocar fusibles externos)	

Descripción de los puertos de comunicación

Descripciones de clavijas Puerto 1		
1	0V	Conexión (-)(GND)
2	5V	Conexión (+)
3	RXD	Recibe datos (RS-232C)
4	TXD	Transmite datos (RS-232C)
5	5V	Conexión (+)
6	0V	Conexión (-)(GND)



Descripciones de clavijas Puerto 2		
1	5V	Conexión (+)
2	TXD	Transmite datos (RS-232C)
3	RXD	Recibe datos (RS-232C)
4	RTS	Ready to send
5	CTS	Clear to send
6	RXD-	Recibe datos (-) (RS-422/485)
7	0V	Conexión (-) (GND)
8	0V	Conexión (-) (GND)
9	TXD+	Transmite datos (+) (RS-422/485)
10	TXD-	Transmite datos (-) (RS-422/485)
11	RTS+	Ready to send (+) (RS-422/485)
12	RTS-	Ready to send (-) (RS-422/485)
13	RXD+	Recibe datos (+) (RS-422/485)
14	CTS+	Clear to send (+) (RS-422/485)
15	CTS-	Clear to send (-) (RS-422/485)

Comunicaciones del Puerto 1	
Com 1	Se conecta a HPP, DirectSOFT, interfaces de operador, etc.
	6 cavijas, RS232C
	Tasa de communicación(baud): 9600 (fija)
	Paridad: odd (valor original de fábrica)
	Dirección de la estación: 1 (fija)
	8 bits de datos
	1 bit start, 1 bit stop
	Asíncrono, half-duplex, DTE
	Protocolo: (Seleccionable automáticamente) K-sequence (solamente esclavo), DirectNET (solamente esclavo), MODBUS (solamente esclavo)

Comunicaciones del Puerto 2	
Com 2	Se conecta a HPP, DirectSOFT, interfaces de operador, etc.
	15-clavijas, puerto de funciones múltiples, RS232C, RS422, RS485
	Tasa de comunicación (baud): 300, 600, 1200, 2400, 4800, 9600, 19200, 38400
	Paridad: odd (valor original), even, 0 (nada)
	Dirección de la estación: 1 (valor original)
	8 bits de datos
	1 bit start, 1 bit stop
	Asíncrono, half-duplex, DTE
	Protocolos: (selección automática) K-sequence (solamente esclavo), DirectNET (maestro/esclavo), MODBUS (maestro/esclavo), non-sequence/print/ASCII in/out

Especificaciones de la CPU

Especificaciones	
Característica	**Detalle**
Memoria total de programa (palabras)	14.8K
Memoria Ladder (palabras)	7680
Memoria V total	7616
Memoria V de usuario (palabras)	7488
Memoria V no volátil (palabras)	128
Tiempo de ejecución de un contacto	0,6 us
Tiempo de barrido típico para 1K booleano	1 - 2 ms
Programación de estilo ladder RLL	Si
Programación RLL y RLLPLUS	Si
Modificaciones del programa durante el modo RUN	Si
Tiempo de barrido	Variable o fijo
Programador portátil	Si
Programación con *DirectSOFT* para Windows	Si
Puertos de comunicación incluídos (RS232C)	Si
Memoria FLASH	Normal en la CPU
Puntos de E/S discretos disponibles locales	36
Canales locales de E/S análogas, máximo	Ninguno
E/S HSIO(cuadratura, tren de pulsos, interrupción, etc.)	Si, 2
Cantidad de puntos de entradas y salidas locales	20 entradas, 16 salidas
Cantidad de instrucciones disponibles	229
Relevadores de control internos	1024
Relevadores especiales (Definidos por el sistema)	512
Etapas en RLLPLUS	1024
Temporizadores	256
Contadores	128
Entradas y salidas inmediatas	Si
Entradas de interrupción (externas o por tiempo)	Si
Subrutinas	Si
Lazos de For/Next	Si
Operaciones aritméticas (Con enteros y con punto flotante)	Si
Instrucciones de secuenciador de tambor (Drum)	Si
Hora y fecha	Si
Diagnóstico interno	Si
Seguridad con contraseña	Si
Registro de errores del sistema	Si
Registro de errores del usuario	Si
Respaldo por batería	D2-BAT-1 opcional disponible (no viene incluída con el PLC)

ANEXO B. ESPECIFICACIONES TECNICAS DEL LATARGETA DE CONTROL ARDUINO MEGA

Resumen especificaciones técnicas ARDUINO MEGA 1280.

Microcontrolador	ATmega1280
Voltaje de funcionamiento de E/S	5V
Voltaje de alimentación (recomendado)	7-12V
Voltaje de alimentación (límite)	6-20V
Pines E/S digitales	54 (14 proporcionan salida PWM)
Pines de entrada analógica	16
Max. Corriente por pin	40 mA
Max. Corriente en el pin 3.3V	50 mA
Memoria Flash	128 KB de las cuales 4 KB las usa el gestor de arranque (bootloader)
SRAM	8 KB
EEPROM	4 KB

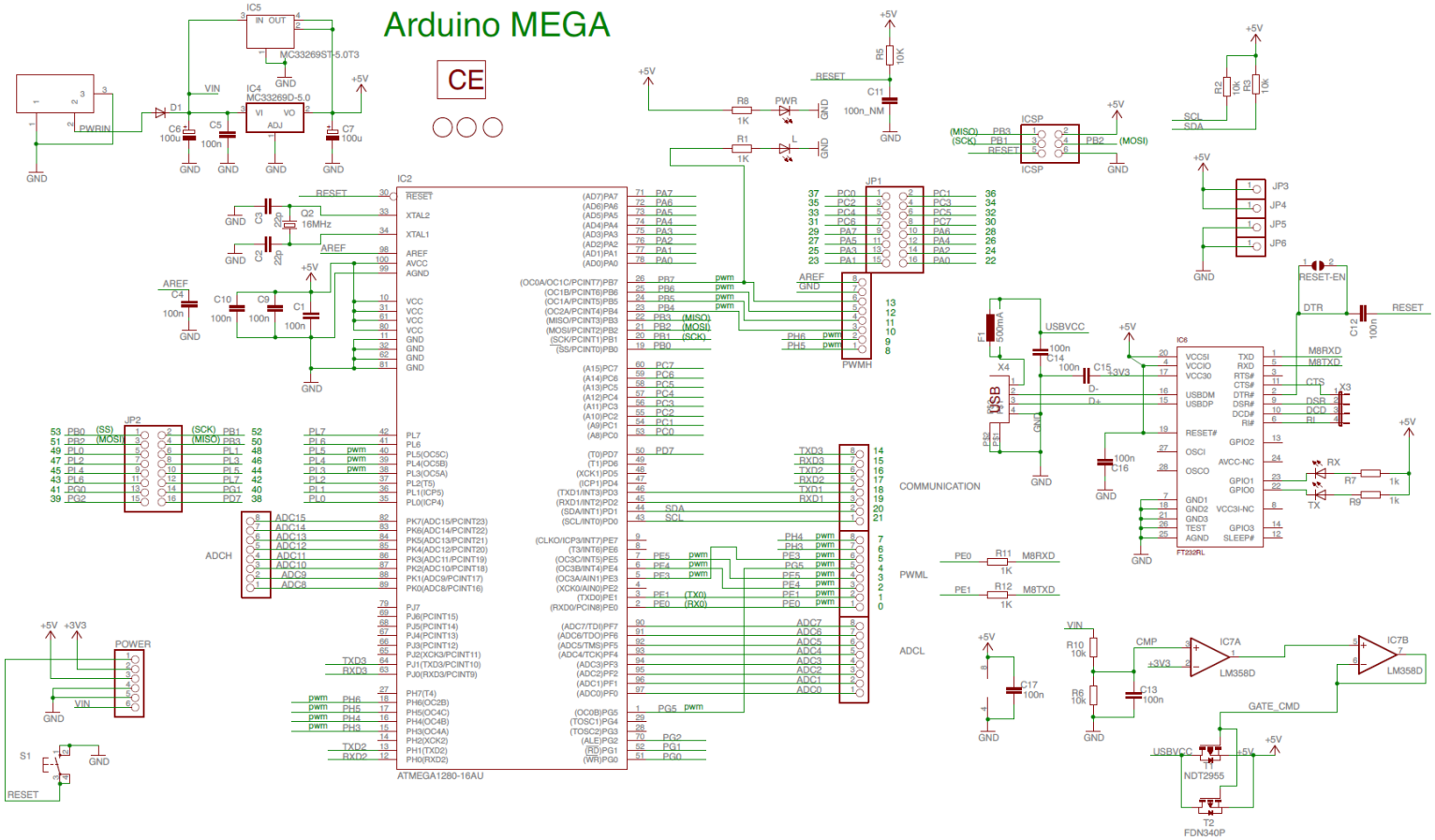
Pines de alimentación ARDUINO MEGA 1280.

PIN	CARACTERÍSTICAS
VIN	La entrada de voltaje a la placa Arduino cando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.
5V	La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
3V3	Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.
GND	Pines de toma de tierra.

Resumen especificaciones técnicas Microcontrolador ATMEL ATMEG 1280.

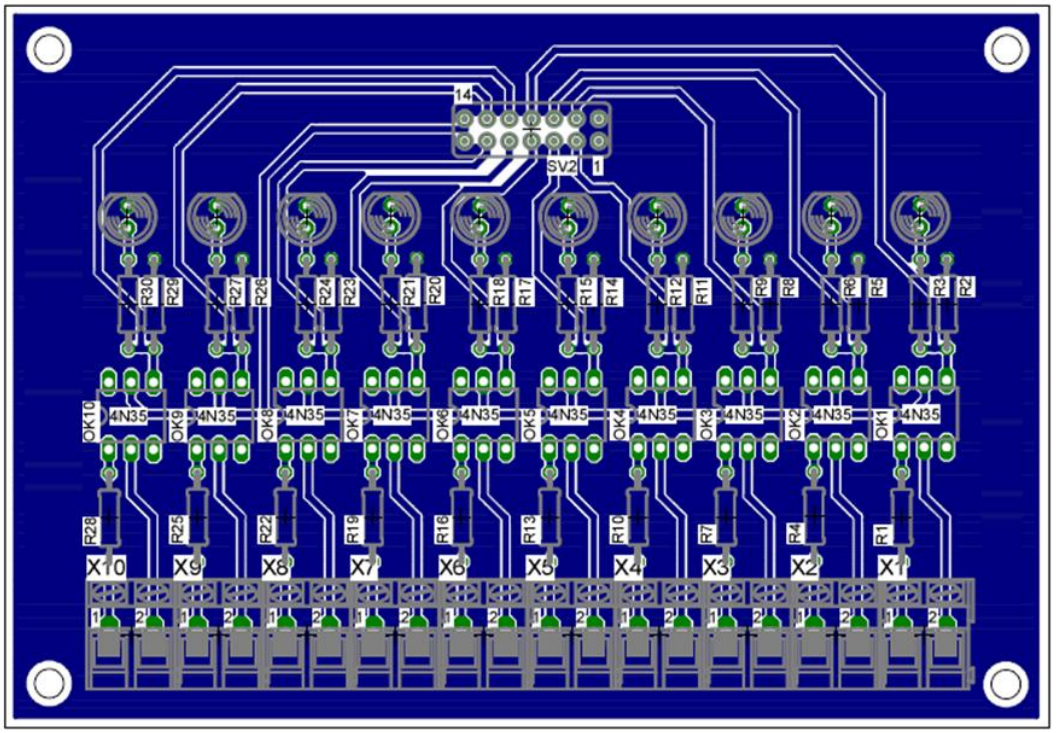
Tamaño del núcleo	8BIT
Tamaño de la memoria de programa	128KB
Tamaño de la memoria EEPROM	4KB
Velocidad de la CPU	16MHZ
Tipo de oscilador	exterior, interior
Voltaje de suministro	2.7V a 5.5V
Rango de temperatura de operación	-40 °C a +85 °C
Número de pines	100
Número de entradas y salidas	86
Tamaño de la memoria Flash	128KB
Interfaz	Interfaz serie a paralelo, Transmisor/ Receptor universal asíncrono

Esquema eléctrico tarjeta de control ARDUINO MEGA

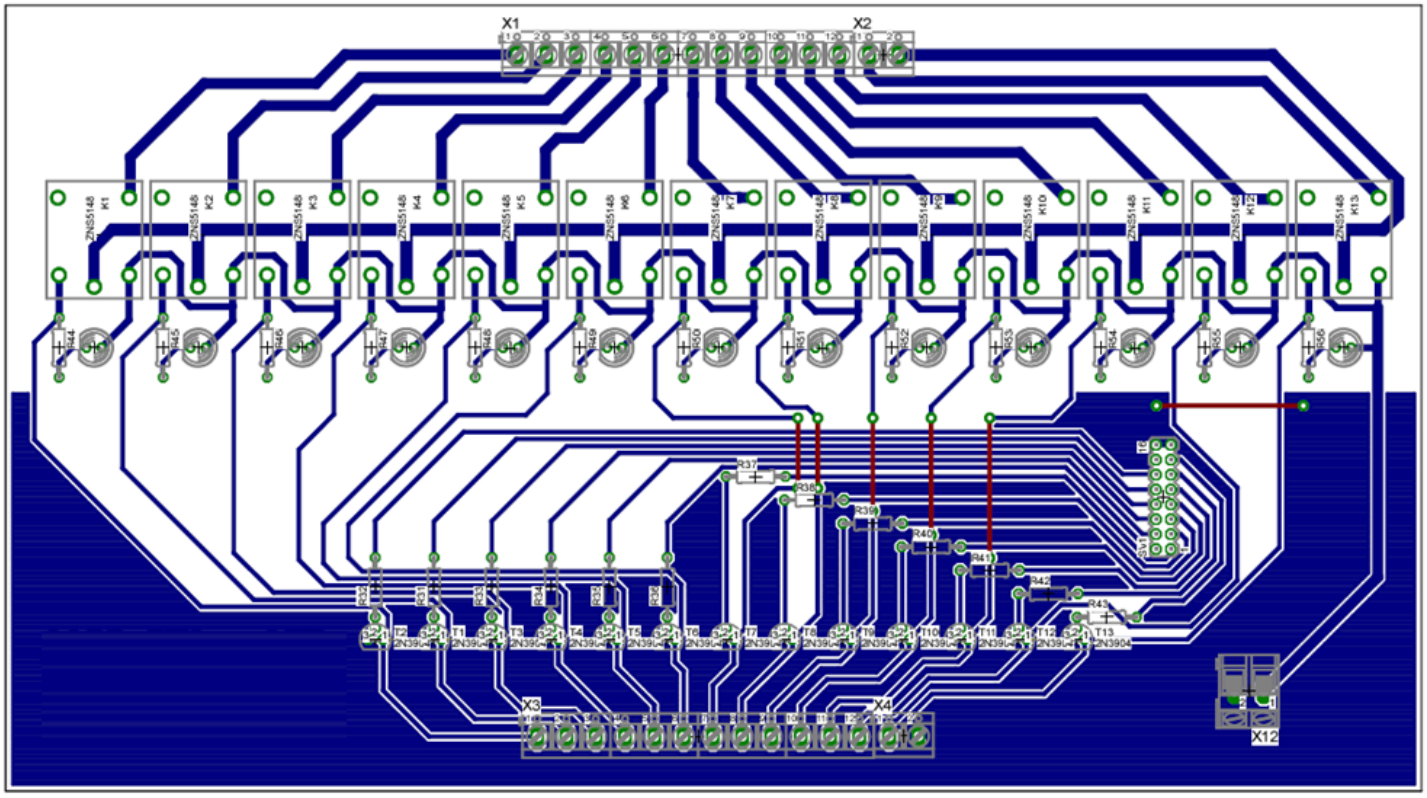


ANEXO C. CURCUITOS IMPRESOS DE LAS TARGETAS.

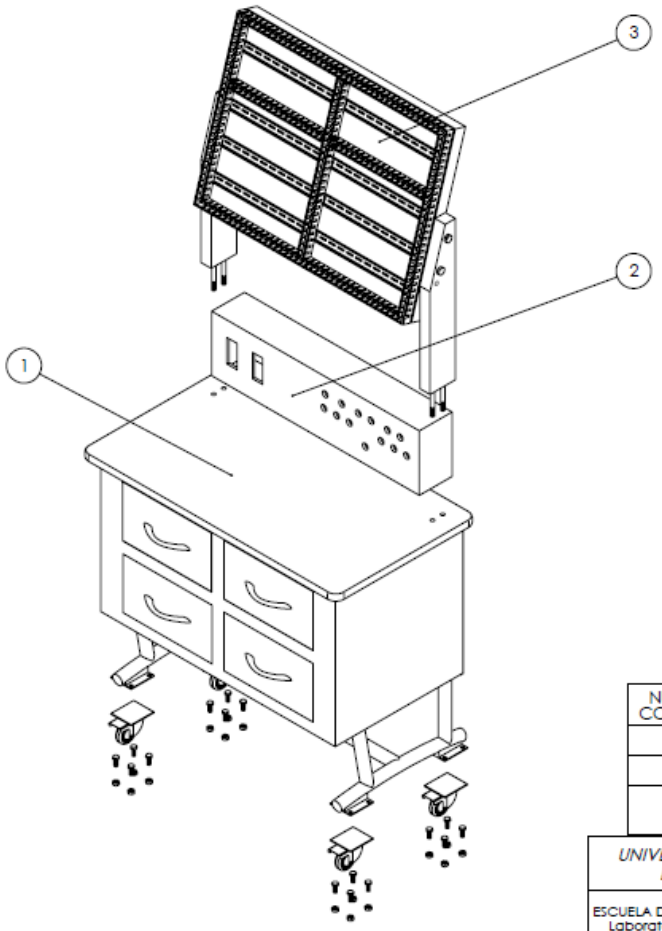
Circuito impreso de la tarjeta de interfaces para entrada de señales.



Circuito impreso de la tarjeta de potencia.

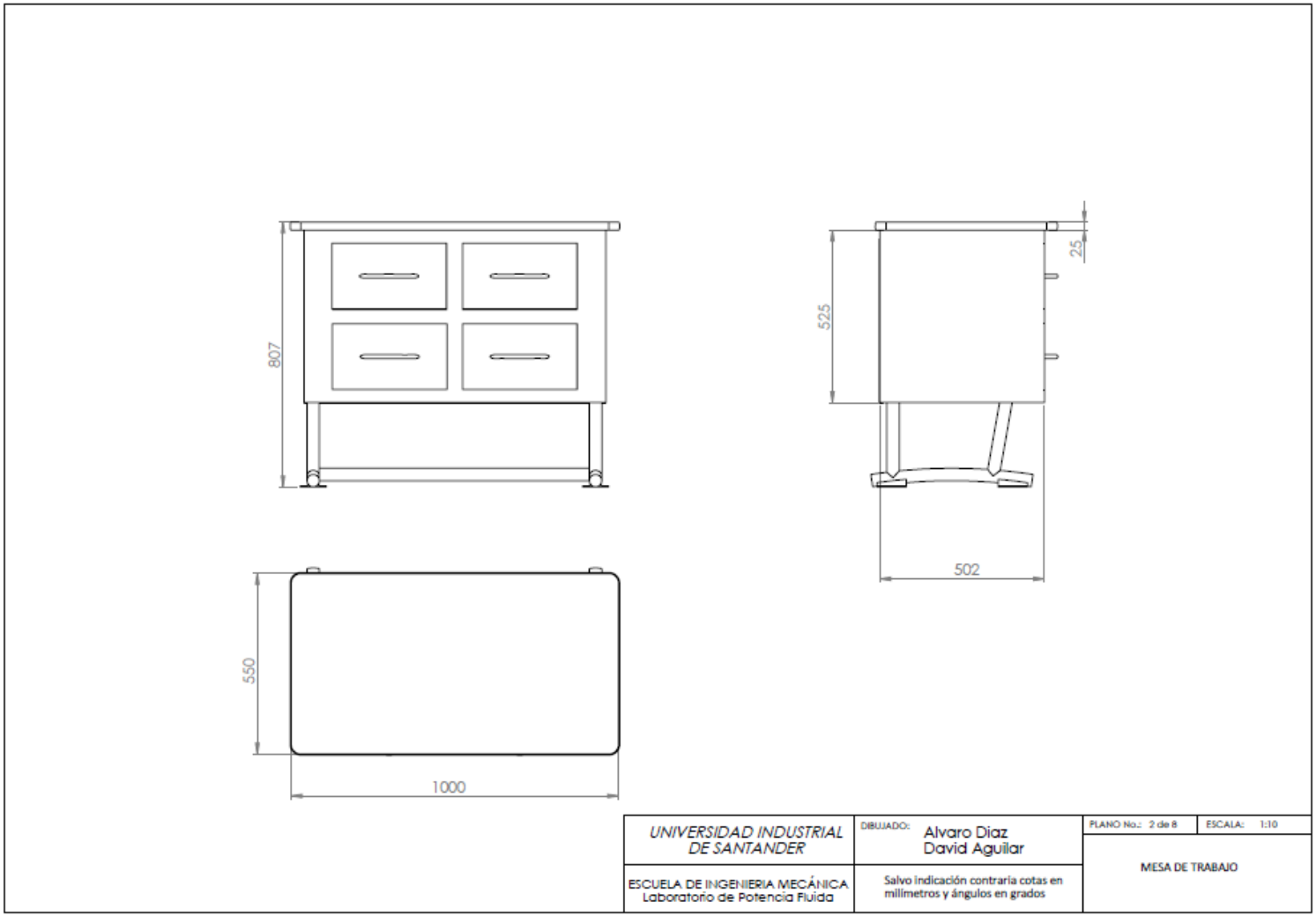


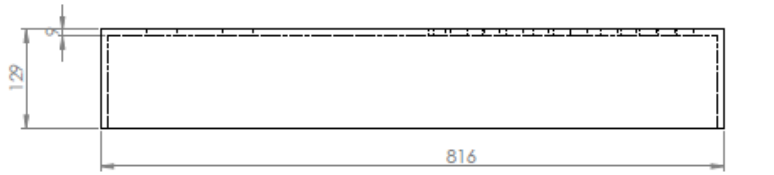
ANEXO D. PLANOS DE LA ESTRUCTURA

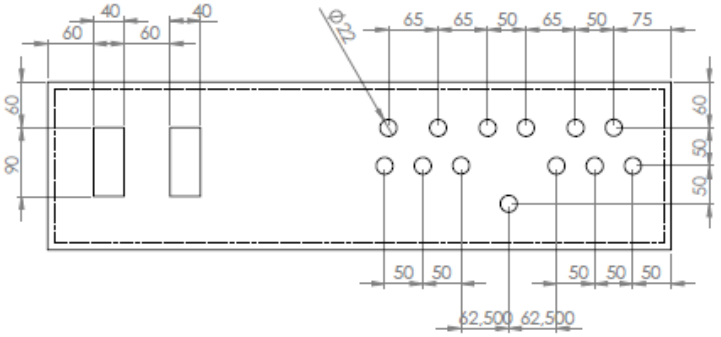


NÚMERO DE COMPONENTE	NÚMERO DE DOCUMENTO	TÍTULO	MATERIAL	CANTIDAD
1	Plano 2	Mesa de trabajo	Acero-Madera	1
2	Plano 3	Tablero Electrico	Madera	1
3	Planos 4 a 8	Panel de Componentes de control	Acero-Madera_plastico	1

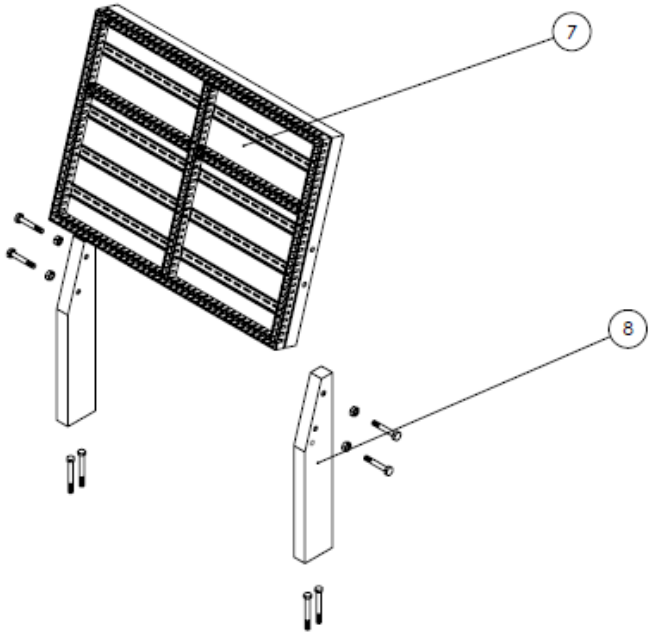
UNIVERSIDAD INDUSTRIAL DE SANTANDER	DIBUJADO: Alvaro Diaz, David Aguilar	PLANO No.: 1 de 8 ESCALA: 1:10
ESCUELA DE INGENIERIA MECÁNICA Laboratorio de Potencia Fluida	Salvo indicación contraria cotas en milímetros y ángulos en grados	BANCO DE SISTEMA DE CONTROL


807
525
25
502
550
1000
UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERIA MECÁNICA
Laboratorio de Potencia Fluida
DIBUJADO: Alvaro Diaz
David Aguilar
Salvo indicación contraria cotas en milímetros y ángulos en grados
PLANO No.: 2 de 8
ESCALA: 1:10
MESA DE TRABAJO



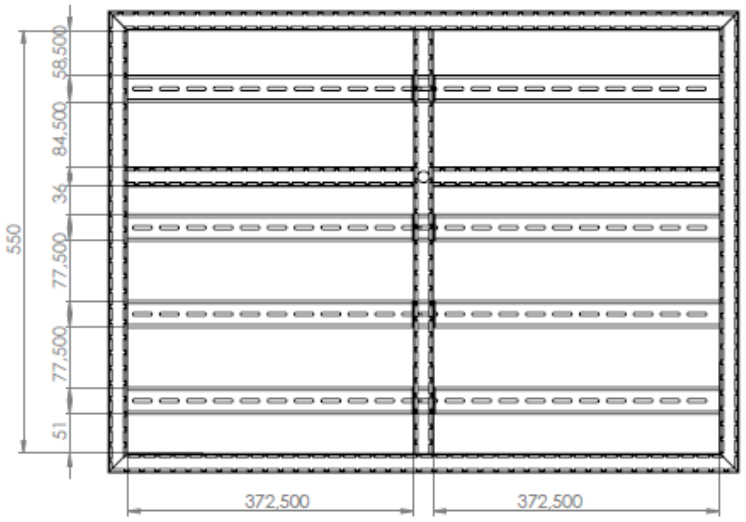


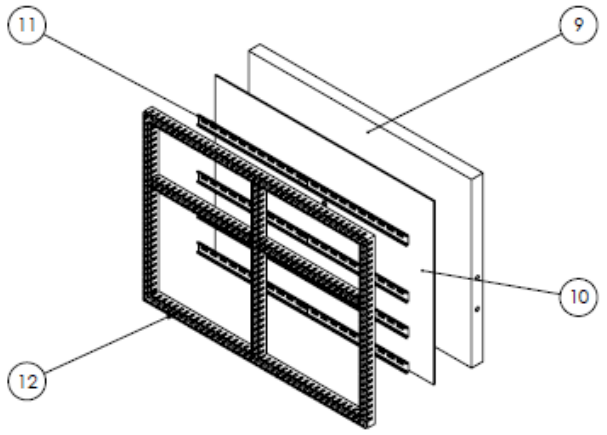
UNIVERSIDAD INDUSTRIAL DE SANTANDER	DIBUJADO: Alvaro Diaz, David Aguilar	PLANO No.: 3 de 8	ESCALA: 1:5
ESCUELA DE INGENIERIA MECÁNICA, Laboratorio de Potencia Fluida	Salvo indicación contraria cotas en milímetros y ángulos en grados	TABLERO ELÉCTRICO	



NÚMERO DE COMPONENTE	NÚMERO DE DOCUMENTO	TÍTULO	MATERIAL	CANTIDAD
7	Planos 5 a 7	Panel de Soporte de elementos de control	Acero-plastico-madera	1
8	Plano 8	Soporte Panel	Acero	2

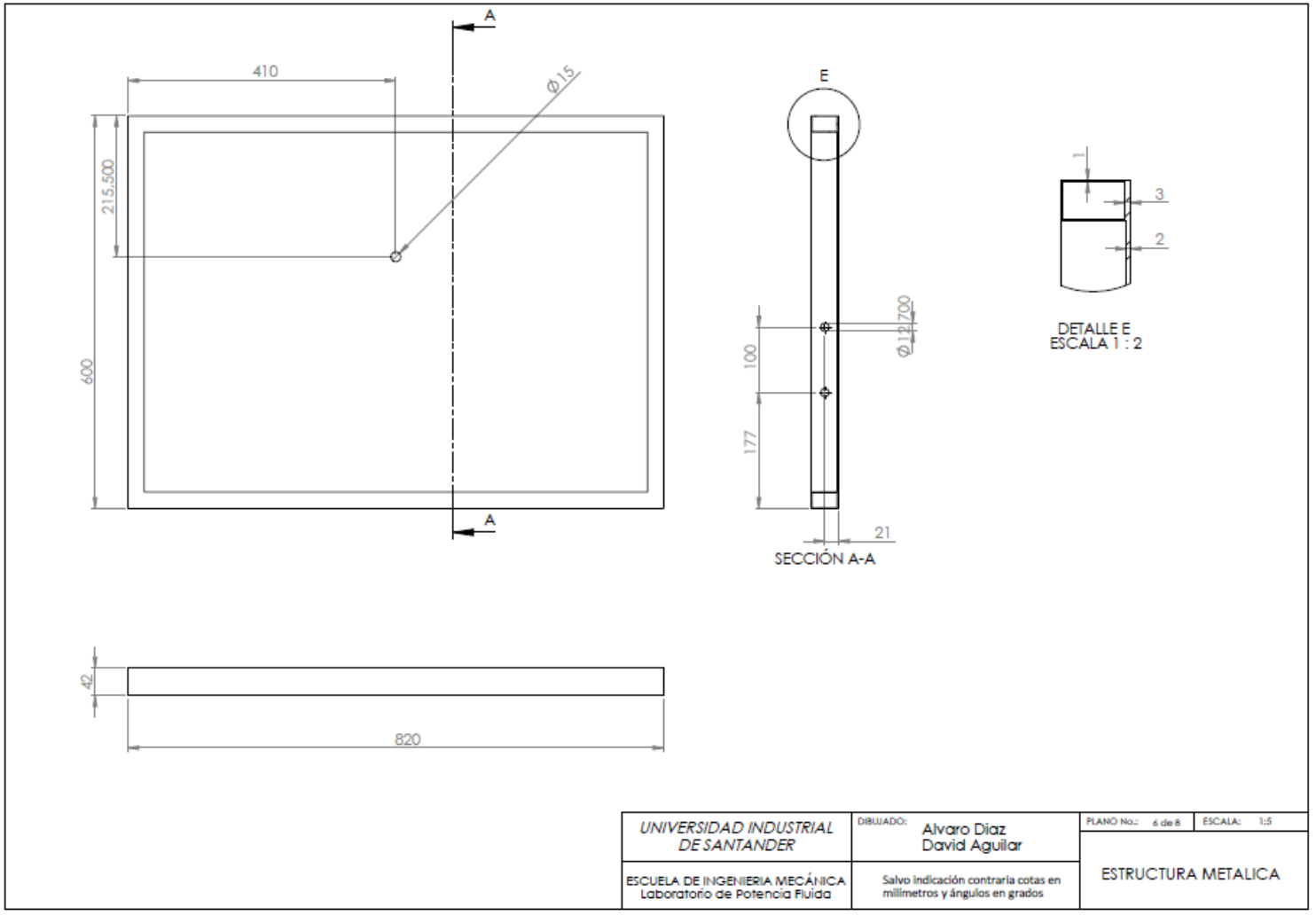
UNIVERSIDAD INDUSTRIAL DE SANTANDER	DIBUJADO: Alvaro Diaz, David Aguilar	PLANO No.: 4 de 8	ESCALA: 1:10
ESCUELA DE INGENIERIA MECÁNICA, Laboratorio de Potencia Fluida	Salvo indicación contraria cotas en milímetros y ángulos en grados	PANEL DE COMPONENTES DE CONTROL	

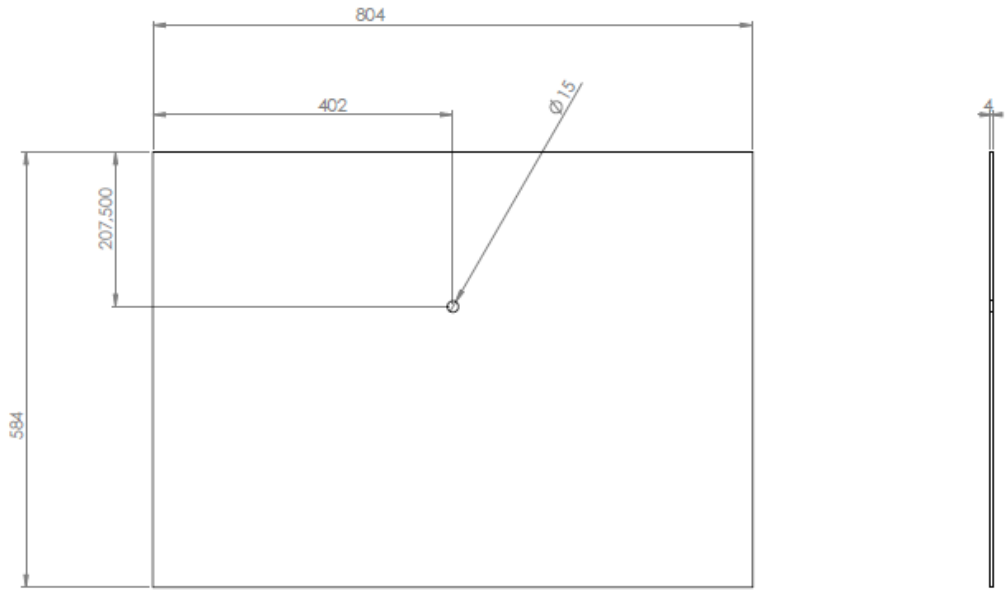




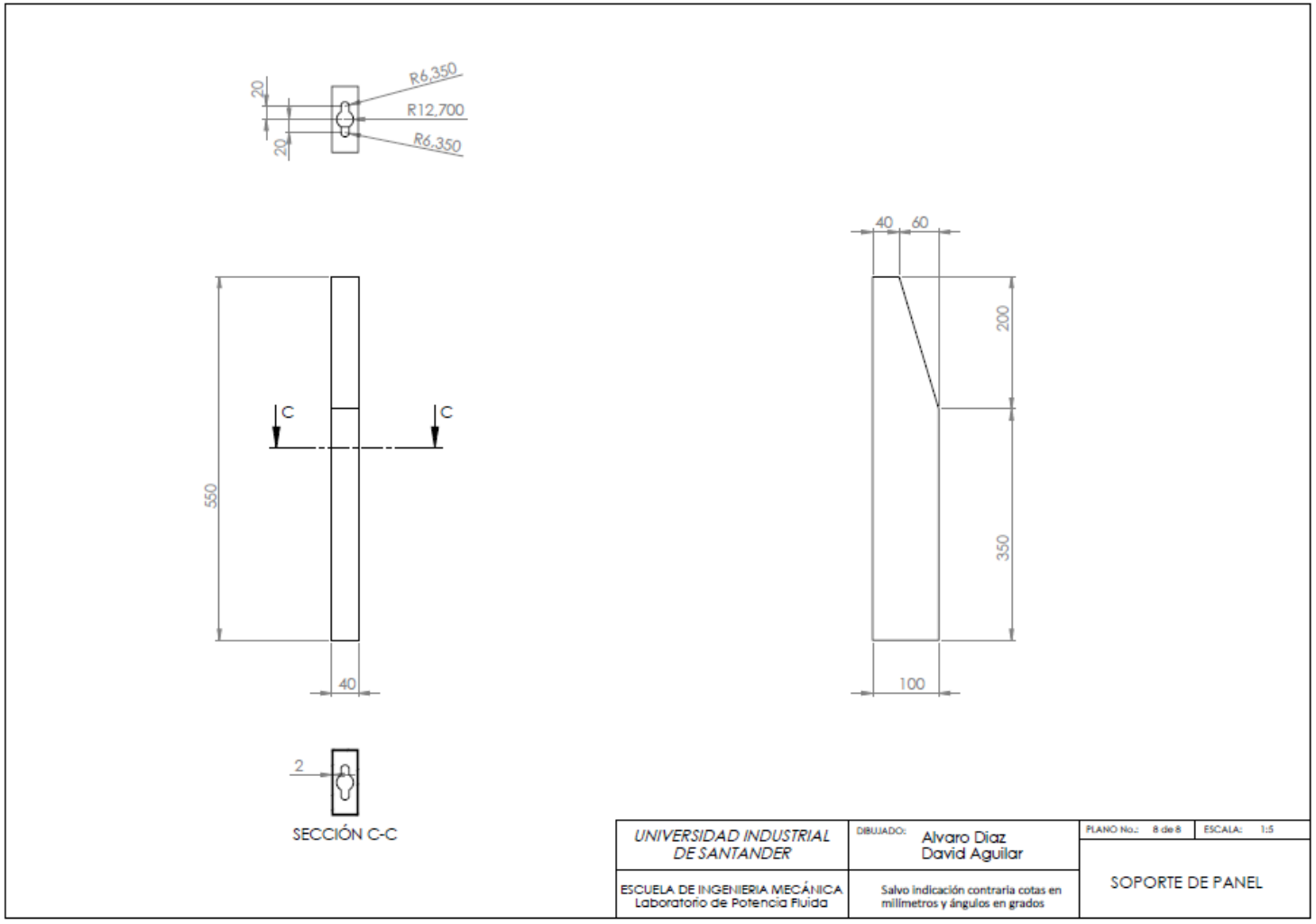
NÚMERO DE COMPONENTE	NÚMERO DE DOCUMENTO	TÍTULO	REFERENCIA	MATERIAL	CANTIDAD
9	Plano 6	Estructura Metálica		Acero	1
10	Plano 7	MDF Recubierto en Fórmica		Madera	1
11		Riel DIN	RD01	Aluminio	4
12		Canaleta Ranurada	3CARGR25x40	Plástico	7

UNIVERSIDAD INDUSTRIAL DE SANTANDER	DIBUJADO: Alvaro Diaz, David Aguilar	PLANO No.: 5 de 7	ESCALA: 1:5
ESCUELA DE INGENIERIA MECÁNICA Laboratorio de Potencia Fluida	Salvo indicación contraria cotas en milímetros y ángulos en grados	PANEL DE SOPORTE DE ELEMENTOS DE CONTROL	


A
410
Ø15
215,500
600
A
E
Ø12,700
100
177
21
SECCIÓN A-A
1
3
2
DETALLE E
ESCALA 1 : 2
42
820
UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERIA MECÁNICA
Laboratorio de Potencia Fluida
DIBUJADO:
Alvaro Diaz
David Aguilar
Salvo indicación contraria cotas en milímetros y ángulos en grados
PLANO No.: 6 de 8
ESCALA: 1:5
ESTRUCTURA METALICA



UNIVERSIDAD INDUSTRIAL DE SANTANDER	DIBUJADO: Alvaro Diaz, David Aguilar	PLANO No.: 7 de 8	ESCALA: 1:5
ESCUELA DE INGENIERIA MECÁNICA, Laboratorio de Potencia Fluida	Salvo indicación contraria cotas en milímetros y ángulos en grados	MDF RECUBIERTA EN FORMICA	


R6,350
R12,700
R6,350
20
20
40
60
200
350
100
550
C
C
40
2
SECCIÓN C-C
UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERIA MECÁNICA
Laboratorio de Potencia Fluida
DIBUJADO:
Alvaro Diaz
David Aguilar
Salvo indicación contraria cotas en milímetros y ángulos en grados
PLANO No.: 8 de 8
ESCALA: 1:5
SOPORTE DE PANEL