ESCUELA POLITÉCNICA DEL EJÉRCITO
SEDE - LATACUNGA

Carrera de Ingeniería Electromecánica

DISEÑO Y CONSTRUCCIÓN DE UN PLC MODULAR A BASE DE MICROCONTROLADORES

ELABORADO POR:

JUAN CARLOS BEDÓN SALAZAR
IVÁN RODRIGO LEÓN GUERRERO

Tesis presentada como requisito parcial para la obtención del grado de:

INGENIERO ELECTROMECÁNICO

Latacunga – Ecuador
Octubre – 2009

ESCUELA POLITÉCNICA DEL EJÉRCITO

CARRERA DE INGENIERÍA ELECTROMECÁNICA

DECLARACIÓN DE RESPONSABILIDAD

Nosotros, Iván Rodrigo León Guerrero y Juan Carlos Bedón Salazar.

DECLARAMOS QUE:

El proyecto de grado denominado "Diseño y Construcción de un PLC Modular a Base de Microcontroladores" ha sido desarrollado en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía. En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Octubre del 2009.

Juan Carlos Bedón Salazar
C.I 0502524978

Iván Rodrigo León Guerrero
C.I 1803187077

ESCUELA POLITÉCNICA DEL EJÉRCITO

CARRERA DE INGENIERÍA ELECTROMECÁNICA

AUTORIZACIÓN

Nosotros, Juan Carlos Bedón Salazar e Iván Rodrigo León Guerrero

Autorizamos a la Escuela Politécnica del Ejército la publicación en la Biblioteca Virtual de la Institución del trabajo "Diseño y Construcción de un PLC Modular a Base de Microcontroladores" cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Latacunga, Octubre del 2009.

Juan Carlos Bedón Salazar
C.I 0502524978

Iván Rodrigo León Guerrero
C.I 1803187077

AGRADECIMIENTO

Nuestro más sincero agradecimiento a la ESCUELA POLITÉCNICA DEL EJÉRCITO, por abrirnos las puertas de tan noble institución para así culminar con nuestras metas e ideales trazados, a nuestros profesores que nos guiaron en el transcurso de nuestra carrera impartiéndonos su sabio conocimiento y brindándonos sobre todo su amistad incondicional, un agradecimiento especial a los Ingenieros, Marco Singaña y Freddy Salazar, por su acertada dirección y recomendaciones durante el lapso del proyecto.

Iván R. León Guerrero.

Juan C. Bedón Salazar.

DEDICATORIA

Son tantas personas a las cuales debo parte de este triunfo, de lograr alcanzar mi culminación académica, la cual es el anhelo de todos los que así lo deseamos.

Definitivamente, Dios, mi Señor, mi Guía, mi Proveedor, mi fin último; sabes lo esencial que has sido en mi posición firme de alcanzar esta meta, esta alegría, que si pudiera hacerla material, la hiciera para entregártela, pero a través de esta meta, podré siempre de tu mano alcanzar otras que espero sean para tu Gloria.

A mis padres, mi hermano, mi cuñada, y mi querida y bella sobrina, por darme la estabilidad emocional, económica, sentimental; para poder llegar hasta este logro, que definitivamente no hubiese podido ser realidad sin ustedes. GRACIAS por darme la posibilidad de que de mi boca salga esa palabra…FAMILIA. Padres, serán siempre mi inspiración para alcanzar mis metas, por enseñarme que todo se aprende, y que todo esfuerzo es al final recompensa. Su esfuerzo, se convirtió en su triunfo y el mío, LOS AMO.

A todos mis amigos pasados y presentes; pasados por ayudarme a crecer y madurar como persona y presentes por estar siempre conmigo, apoyándome en todo las circunstancias posibles, también son parte de esta alegría, LOS RECUERDO.

Y a todos aquellos, que han quedado en los recintos más escondidos de mi memoria, pero que fueron participes de esta alegría que hoy llevo en mi alma.

Iván R. León Guerrero.

DEDICATORIA

Dedico este proyecto toda mi carrera universitaria a Dios por ser quien ha estado a mi lado en todo momento dándome las fuerzas necesarias para continuar luchando día tras día y seguir adelante rompiendo todas las barreras que se me presentaron

A mi mamá Teresa Salazar y mi papá Carlos Bedón ya que gracias a ellos soy quien soy hoy en día, fueron los que me dieron ese cariño y calor humano necesario, son los que ha velado mi salud, mis estudios, mediación a los que debo los consejos para formarme como un ser integral y de las cuales me siento orgulloso

A mis hermanos los cuales han estado a mi lado, han compartido todos los secretos y aventuras que lo puede vivir los hermanos que han estado siempre alerta a cualquier problema que se presenta.

También a todos mis amigos más cercanos, a esos amigos que siempre me acompañaron y con los cuales he contado desde que los conocí

Juan C. Bedón Salazar

CERTIFICACIÓN

Certificamos, que el presente proyecto de grado fue desarrollado en su totalidad por los señores JUAN CARLOS BEDÓN SALAZAR E IVÁN RODRIGO LEÓN GUERRERO, previo a la obtención de su Título de Ingeniero Electromecánico respectivamente.

Latacunga, Octubre del 2009

Ing. Marco Singaña

Ing. Freddy Salazar

Ab. Eduardo Vásquez Alcázar

ÍNDICE GENERAL

CAPÍTULO I

CAPÍTULO V

ÍNDICE DE TABLAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

ÍNDICE FIGURAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

CAPÍTULO IV

CAPÍTULO I

MARCO TEÓRICO

1. INTRODUCCIÓN A LOS PLC.

Hasta no hace mucho tiempo el control de procesos industriales se venía haciendo de forma cableada por medio de contactores y relés. Al operador que se encontraba a cargo de este tipo de instalaciones, se le exigía tener altos conocimientos técnicos para poder realizarlas y posteriormente mantenerlas. Además cualquier variación en el proceso suponía modificar físicamente gran parte de las conexiones de los montajes, siendo necesario para ello un gran esfuerzo técnico y un mayor desembolso económico.

En la actualidad no se puede entender un proceso complejo de alto nivel desarrollado por técnicas cableadas. El ordenador y los autómatas programables han intervenido de forma considerable para que este tipo de instalaciones se hayan visto sustituidos por otros, controlados de forma programada.

El PLC (Programmable Logic Controller) es un equipo electrónico programable diseñado para controlar en tiempo real y en ambiente industrial un proceso secuencial. Se produce una reacción a la información recibida por los sensores del sistema automatizado (finales de carrera, células fotoeléctricas, sensores, encoders, teclados, etc.) y se actúa sobre los accionadores de la instalación (motores, electroválvulas, indicadores luminosos, etc.).

El avance de la automatización ha ido invariablemente unido al avance de los sistemas eléctricos y electrónicos. A medida que se han ido mejorando los sistemas informáticos y reduciendo el tamaño de los componentes electrónicos, se han podido construir autómatas con mayor capacidad de control sobre los sistemas, se ha reducido su tamaño y aumentado sus posibilidades.

El "núcleo" o "cerebro" del autómata es la CPU, en torno a la cual se añaden los distintos módulos que completan la característica de la "caja" del autómata.

Existen muchos módulos diferentes, adaptables para cada familia de autómatas, siendo los más comunes los módulos de E/S analógicas y digitales, el módulo de la fuente de tensión e intensidad, entradas de interrupción, módulos de contadores, de puertos, etc.

Antes de la aparición del PLC, existían diversos problemas con los relés, que cuando los requerimientos de producción cambiaban también lo hacía el sistema de control. Esto comenzó a resultar bastante costoso cuando los cambios fueron frecuentes. Dado que los relés son dispositivos mecánicos y poseen una vida limitada, se requería un estricto mantenimiento. Por otra parte, a veces se debían realizar conexiones entre cientos o miles de relés, lo que implicaba un enorme esfuerzo de diseño y mantenimiento.

Los "nuevos controladores" debían ser fácilmente programables por ingenieros de planta o personal de mantenimiento. El tiempo de vida debía ser largo y los cambios en el programa tenían que realizarse de forma sencilla. Finalmente se imponía que debían trabajar sin problemas en entornos industriales adversos. La solución fue el empleo de una técnica de programación familiar y el reemplazo de los relés mecánicos por relés de estado sólido.
En la Figura 1.1, se presenta el esquema físico de un PLC.

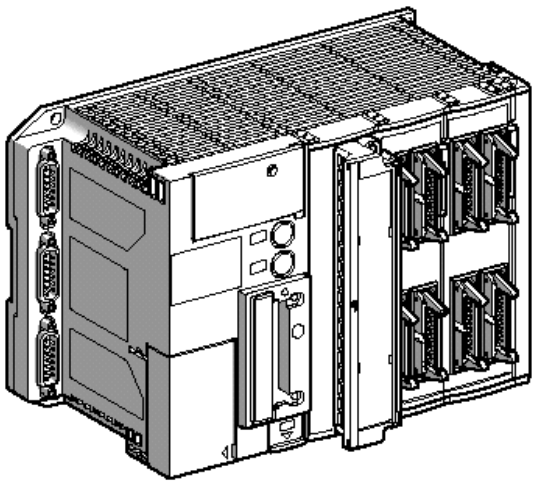


Figura 1. 1 Esquema físico del PLC.

A mediados de los 70 las tecnologías dominantes de los PLC eran máquinas de estado secuencial y CPU basadas en desplazamiento de bit. Los microprocesadores convencionales dieron la potencia necesaria para resolver de forma rápida y completa la

lógica de los pequeños PLC's. Por cada modelo de microprocesador, había un modelo de PLC basado en el mismo.

Las habilidades de comunicación comenzaron a aparecer en 1973 aproximadamente. El primer sistema fue el Bus Modicon (Modbus)[1]. El PLC podía ahora comunicarse con otros PLC's y en conjunto podían estar aislados de las máquinas que controlaban. También podían enviar y recibir señales de tensión variables, entrando en el mundo analógico. Desafortunadamente, la falta de un estándar, acompañado de un continuo cambio tecnológico ha hecho que la comunicación de PLC's tengan sistemas físicos y protocolos incompatibles entre sí.

En los 80 se produjo un intento de estandarización de las comunicaciones, también fue un tiempo en el que se redujeron las dimensiones del PLC y se pasó a programar con programación simbólica a través de ordenadores personales, en vez de los clásicos terminales de programación. Hoy día el PLC más pequeño es del tamaño de un simple relé.

Los 90 han mostrado una gradual reducción en el número de nuevos protocolos, el último estándar (IEC 1131-3) intenta unificar el sistema de programación de todos los PLC en un único estándar internacional. Ahora se dispone de PLC's que pueden ser programados en diagramas de bloques, lista de instrucciones y texto estructurado al mismo tiempo.

Los PC están comenzando a reemplazar al PLC en algunas aplicaciones, incluso la compañía que introdujo el Modicon 084 ha cambiado al control basado en PC. Por lo cual, no sería de extrañar que en un futuro no muy lejano el PLC desaparezca frente al cada vez más potente PC, debido a las posibilidades que éstos pueden proporcionar.

1.1 PARTES DE UN AUTOMATA PROGRAMABLE.

La estructura básica de cualquier autómata es la siguiente:

- Fuente de alimentación
- CPU

[1] http://www.uclm.es/profesorado/rcarcelen_plc/control.htm

- Módulo de entrada
- Módulo de salida
- Terminal de programación
- Periféricos.

Respecto a su disposición externa, los autómatas pueden contener varias de estas secciones en un mismo módulo o cada una de ellas separadas por diferentes módulos. Así, se pueden distinguir autómatas Compactos y Modulares.

1.1.1 Fuente de Alimentación.

Es la encargada de convertir la tensión de la red, de 120V o 220V c.a., a baja tensión de c.c, normalmente 24 V. Siendo ésta la tensión de trabajo en los circuitos electrónicos que forma el Autómata.

1.1.2 CPU.

La Unidad Central de Procesos es el auténtico cerebro del sistema. Se encarga de recibir las órdenes del operador por medio de la consola de programación y el módulo de entradas. Posteriormente las procesa, para luego enviar respuestas al módulo de salidas. En su memoria se encuentra residente el programa destinado a controlar el proceso.

1.1.3 Módulo de Entradas.

A este módulo se unen eléctricamente los captadores (interruptores, finales de carrera, pulsadores, etc).

La información recibida en él, es enviada a la CPU para ser procesada de acuerdo a la programación residente.

En las Figuras 1.2 y 1.3 se pueden diferenciar dos tipos de captadores conectados al módulo de entradas: los Pasivos y los Activos.

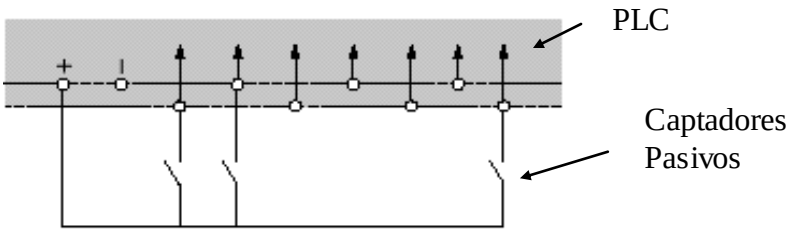


Figura 1. 2 Captadores pasivos.

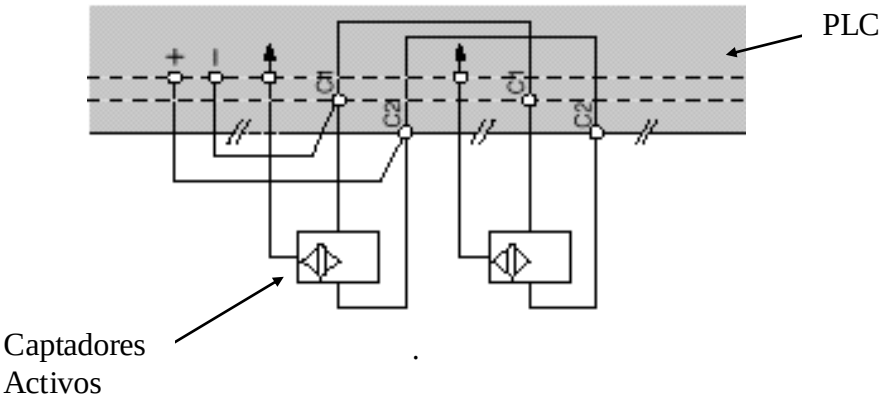


Figura 1. 3 Captadores activos.

1.1.3.1 Captadores Pasivos: Son aquellos que cambian su estado lógico, activado / no activado, por medio de una acción mecánica. Estos son los interruptores, pulsadores, finales de carrera, etc.

1.1.3.2 Captadores Activos: Son dispositivos electrónicos que necesitan ser alimentados por una tensión para que varíen su estado lógico. Este es el caso de los diferentes tipos de detectores (Inductivos, Capacitivos, Fotoeléctricos). Muchos de estos aparatos pueden ser alimentados por la propia fuente de alimentación del autómata.

El operador que conoce circuitos de automatismos industriales realizados por contactores, sabrá que puede utilizar, como captadores, contactos normalmente abiertos o normalmente cerrados, dependiendo de su función en el circuito. Como ejemplo se puede ver un simple arrancador paro/marcha (Figura 1.4). En él se distingue el contacto usado como pulsador de marcha, que es normalmente abierto y el usado como pulsador de parada, que es normalmente cerrado.
Sin embargo en circuitos automatizados por autómatas, los captadores son generalmente abiertos.

El mismo arrancador paro/marcha realizado con un autómata, se presenta en la Figura 1.5, en él se ve que ambos pulsadores y el relé térmico auxiliar son abiertos.

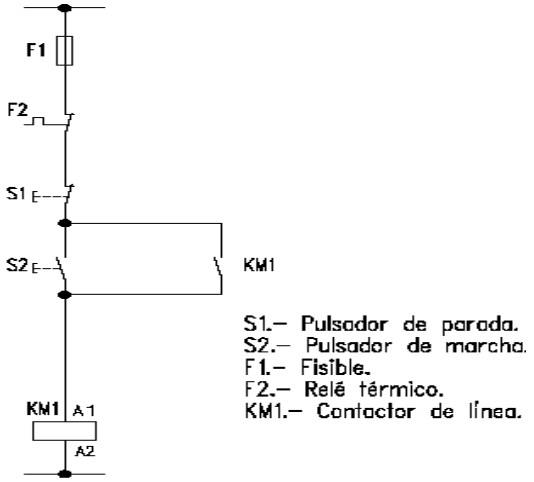


Figura 1. 4 Arrancador paro / marcha.

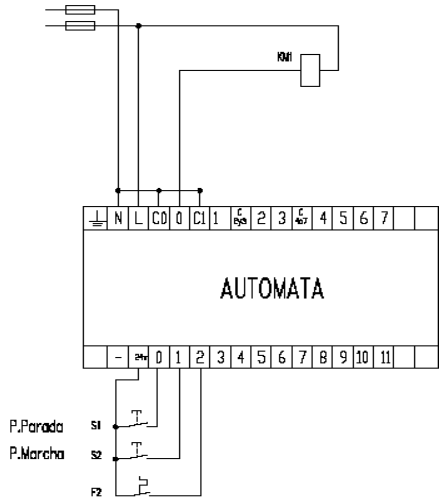


Figura 1. 5 Arrancador paro / marcha con autómata.

1.1.4 Módulos de Salidas.

El módulo de salidas del autómata es el encargado de activar y desactivar los actuadores (bobinas de contactores, lámparas, motores pequeños, etc).

La información enviada por las entradas a la CPU, una vez procesada, se envía al módulo de salidas para que éstas sean activadas y a la vez los actuadores que en ellas están conectados.

Según el tipo de proceso a controlar por el autómata, se puede utilizar diferentes módulos de salidas, existen tres tipos bien diferenciados:

- A relés
- A triac.
- A transistores.

1.1.4.1 Módulos de Salidas a Relés.

Son usados en circuitos de corriente continua y alterna. Están basados en la conmutación mecánica, por la bobina del relé, de un contacto eléctrico normalmente abierto. Este módulo se lo puede observar en la Figura 1.6.

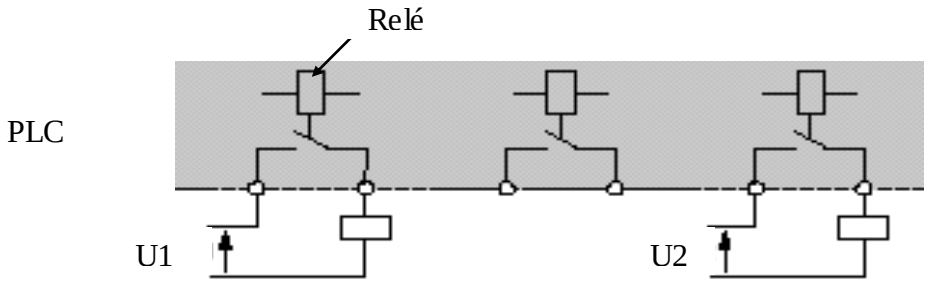


Figura 1. 6 Salidas a relés.

1.1.4.2 Módulos de Salidas a Triacs.

Se utilizan en circuitos de corriente continua y corriente alterna que necesiten maniobras de conmutación muy rápidas. En la Figura 1.7 se representa el módulo de salida a triacs.

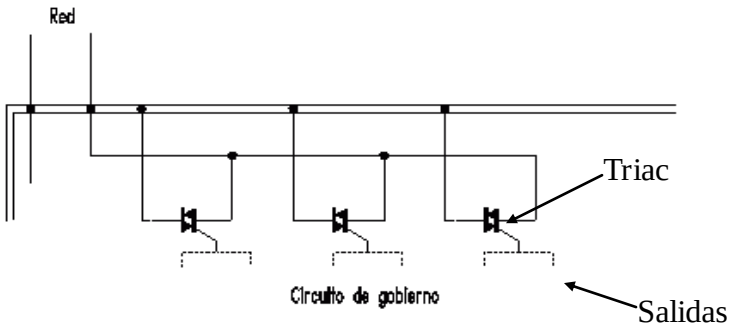


Figura 1. 7 Salidas a triacs.

1.1.4.3 Módulos de Salidas a Transistores a Colector Abierto.

El uso de este tipo de módulos es exclusivo de los circuitos de c.c. Igual que en los de triacs, es utilizado en circuitos que necesiten maniobras de conexión/desconexión muy rápidas.
En la Figura 1.8, se representa el módulo de salida a transistores a colector abierto.

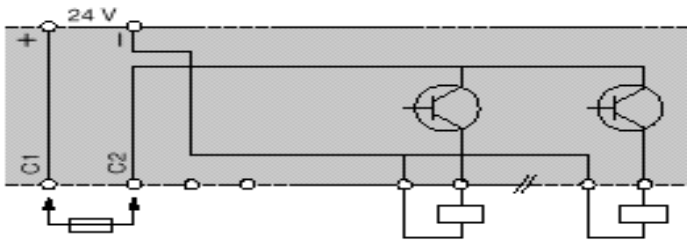


Figura 1. 8 Salida a transistores - colector abierto.

1.1.5 Terminal de Programación.

El terminal o consola de programación es el que permite comunicar al operario con el sistema.

Las funciones básicas de éste, son las siguientes:
- Transferencia y modificación de programas.
- Verificación de la programación.
- Información del funcionamiento de los procesos.

Como consolas de programación pueden ser utilizadas las construidas específicamente para el autómata, tipo calculadora; o bien un ordenador personal, PC, que soporte un software especialmente diseñado para resolver los problemas de programación y control.

1.2 FUNCIONES BÁSICAS DEL PLC.

- **Detección.**

Lectura de la señal de los diferentes pulsadores, sensores, finales de carrera, etc, que están distribuidos en un proceso de fabricación.

- **Mando.**

Elabora y envía las acciones al proceso mediante los accionadores y preaccionadores.

- **Dialogo Hombre – Máquina.**

Mantener un diálogo con los operarios de producción, obedeciendo sus consignas e informando del estado del proceso.

- **Programación.**

Para introducir, elaborar y cambiar el programa de aplicación del autómata. El diálogo de programación debe permitir modificar el programa, incluso con el autómata controlando el proceso.

1.2.1 Nuevas Funciones.

- **Redes de comunicación.**

Las redes industriales permiten la comunicación y el intercambio de datos entre autómatas en tiempo real. En unos cuantos milisegundos pueden enviarse datos e intercambiar tablas de memoria compartida.

- **Sistemas de supervisión.**

También los autómatas permiten comunicarse con ordenadores provistos de programas de supervisión industrial. Esta comunicación se realiza por una red industrial o por medio de una conexión por el puerto serie del ordenador.

- **Control de procesos continuos.**

Además de dedicarse al control de sistemas de eventos discretos, los autómatas llevan incorporadas funciones que permiten el control de procesos continuos. Disponen de módulos de entrada y salida analógicas, y la posibilidad de ejecutar reguladores PID que están programados en el autómata.

- **Entradas- Salidas distribuidas.**

Los módulos de entrada / salida no tienen por qué estar en el armario del autómata. Pueden estar distribuidos por la instalación, se comunican con la unidad central del autómata mediante un cable de red.

- **Buses de campo.**

Mediante un solo cable de comunicación se pueden conectar al bus captadores y accionadores, reemplazando al cableado tradicional. El autómata consulta cíclicamente el estado de los captadores y actualiza el estado de los accionadores.

1.3 TIPOS Y SELECCIÓN DE PLC.

Estas posibilidades suelen estar condicionadas por el número de terminales entrada/salida que posea el autómata. Según esto, los autómatas se pueden clasificar en:

- Grandes.- Con más de 512 terminales de E/S
- Medianos.- Hasta 512 terminales de E/S
- Pequeños.- Hasta 128 terminales de E/S

1.4 INTRODUCCIÓN A LOS MICROCONTROLADORES.

Existen varias empresas que actualmente se encuentran fabricando estas micromputadoras, las cuales poseen variedad de modelos que facilitan la implementación de proyectos que se pueden realizar[2].

Los microcontroladores están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los PC's, en los teléfonos, en los hornos microondas y los televisores de nuestro hogar.

1.4.1 El Microcontrolador.

Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador.
Un microcontrolador dispone normalmente de los siguientes componentes:

- Procesador o UCP (Unidad Central de Proceso).
- Memoria RAM, para contener los datos.
- Memoria para el programa, tipo ROM, PROM ó EPROM.
- Líneas de E/S, para comunicarse con el exterior.

[2] http://www2.ing.puc.cl/~iee3912/files/PIC.pdf

- Diversos módulos para el control de periféricos (temporizadores, Puertos Serie y Paralelo, ADC: Conversores Analógico/Digital, DAC: Conversores Digital/Analógico, etc.)
- Generador de impulsos de reloj, que sincronizan el funcionamiento de todo el sistema.

Las características que incorpora un microcontrolador, permiten disponer de las siguientes ventajas:

Aumento de prestaciones: Un mayor control sobre un determinado elemento representa una mejora considerable en el mismo.
Aumento de la fiabilidad: Al reemplazar el microcontrolador por un elevado número de elementos, disminuye el riesgo de averías y se precisan menos ajustes.
Reducción del tamaño: La integración del microcontrolador en un chip disminuye el volumen, la mano de obra y los stocks.
Mayor flexibilidad: Las características de control están programadas, por lo que para su modificación sólo necesita cambios en el programa de instrucciones.

El microcontrolador es en definitiva un circuito integrado que incluye todos los componentes de un PC. Debido a su reducido tamaño es posible montar el controlador en el propio dispositivo al que gobierna; en este caso el controlador recibe el nombre de controlador empotrado (embedded controller).

Si se dispusiese de un solo modelo de microcontrolador, éste debería tener muchos recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. Esta potenciación supondría en muchos casos un despilfarro.

En la práctica cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más poderosos. Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado, es la selección del microcontrolador a utilizar.

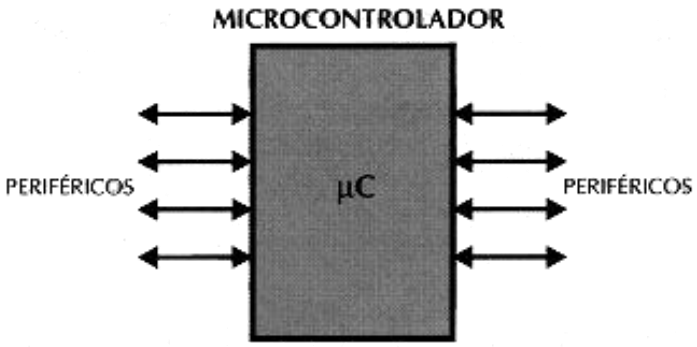


Figura 1. 9 Periféricos de un microcontrolador.

En la Figura 1.9 se observa que el microcontrolador es una caja cerrada. Todas las partes del dispositivo están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.

1.4.2 Aplicaciones de los Microcontroladores.

Cada vez existen más recursos que incorpora un microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y coste, mejorar su fiabilidad y disminuir el consumo.

Algunos fabricantes de microcontroladores superan el millón de unidades de un modelo determinado producidas en una semana. Este dato puede dar una idea de la masiva utilización de estos componentes[3].

Los microcontroladores están siendo empleados en multitud de sistemas presentes en la vida diaria, como pueden ser juguetes, horno microondas, frigoríficos, televisores, computadoras, impresoras, módems, el sistema de arranque de un carro, etc. Estos pequeños microcontroladores podrían comunicarse entre ellos y con un procesador central, probablemente más potente, para compartir la información y coordinar sus acciones; como de hecho, ocurre ya habitualmente con el PC.

1.4.3 El Mercado de los Microcontroladores.

Existe una gran diversidad de microcontroladores. Quizá la clasificación más importante sea entre microcontroladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los

[3] http://www.monografias.com/trabajos12/microco/microco.shtml

microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer.

La razón de esta tendencia es que los microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones[4], lo que hace absurdo emplear microcontroladores más potentes y consecuentemente más caros. En cuanto a las técnicas de fabricación, prácticamente la totalidad de los microcontroladores actuales se fabrican con tecnología CMOS 4 (Complementary Metal Oxide Semiconductor), esta tecnología supera a las técnicas anteriores por su bajo consumo y alta inmunidad al ruido[5].

La distribución de las ventas según su aplicación es la siguiente[6]:

Una tercera parte se absorbe en las aplicaciones relacionadas con los computadores y sus periféricos.
La cuarta parte se utiliza en las aplicaciones de consumo (electrodomésticos, juegos, TV, vídeo, etc.).
El 16% de las ventas mundiales se destinó al área de las comunicaciones.
Otro 16% fue empleado en aplicaciones industriales.
El resto de los microcontroladores vendidos en el mundo, aproximadamente un 10% fueron adquiridos por las industrias de automoción.

También los modernos microcontroladores de 32 bits van afianzando sus posiciones en el mercado, siendo las áreas de más interés el procesamiento de imágenes, las comunicaciones, las aplicaciones militares, los procesos industriales y el control de los dispositivos de almacenamiento masivo de datos.

1.5 TIPOS DE MICROCONTROLADORES.

1.5.1 Microcontrolador ALTAIR[7].

ALTAIR es el nombre genérico de una familia de microcontroladores de propósito general compatibles con la familia MCS 51. Todos ellos son programables directamente desde un

[4] http://lc.fie.umich.mx/~jrincon/intro-microcon1.pdf
[5] http://www.etse.urv.es/~mgarciaf/micro/mcIntro04bn.pdf
[6] http://foro.portalhacker.net/index.php/toPIC,11056.0.html
[7] http://html.rincondelvago.com/microcontroladores.htm

equipo PC mediante el lenguaje macroensamblador, o bien mediante otros lenguajes disponibles para la familia MCS 51 (BASIC, C).

Los microcontroladores ALTAIR disponen de un microprocesador de 8 bits, 100% compatible a nivel de código, 256 bytes de memoria interna, 128 registros especiales de función, puertos de entrada/salida de propósito general, 111 instrucciones y posibilidad de direccionar 128 Kbytes.

Unos microcontroladores ALTAIR se diferencian de otros por el número de entradas salidas, periféricos (DAC, ADC, WATCHDOG, PWM, velocidad de ejecución, etc.); por lo que la elección de un modelo u otro dependerá de las necesidades. Como entrenador o sistema de iniciación se recomienda la utilización de un ALTAIR 32 BASICO o bien un ALTAIR 535A completo. En proyectos avanzados o desarrollos profesionales puede ser preferible un ALTAIR 537 A.

1.5.2 Microcontrolador INTEL[8].

El MCS 8051 es el primer microcontrolador de la familia introducida por Intel Corporation. La familia MCS 8051 de microcontroladores son de 8 bits, capaces de direccionar hasta 64 kbytes de memoria de programa y 64 kbytes de memoria de datos. Los bloques separados de memoria de código y de datos se denominan como la Arquitectura Hardware. El MCS 8051 tiene dos señales de lectura separadas, los pines P3/7RD, (pin 17) y PSEN (pin 29) como se observa en la Figura 1.10. El primero es activado cuando un byte va ser leído desde memoria de datos externo; el otro, cuando un byte va ser leído desde memoria de programa externo. Ambas señales son activadas en nivel bajo; esto es, son reseteados a nivel lógico 0 cuando están activados. Todo código externo es buscado desde memoria de programa externo; además, los bytes de memoria de programa externo pueden ser leídos por instrucciones de lectura especiales, tal como la instrucción MOVC. Hay también instrucciones separadas para leer desde memoria de datos externo, tal como la instrucción MOVX. Esto significa que las instrucciones determinan que bloque de memoria es direccionado, y la señal de control correspondiente, o RD o PSEN, es activado durante el ciclo de lectura de memoria. Un único bloque de memoria puede ser mapeado para actuar como memoria de datos y de programa; esto es lo que se llama la arquitectura

[8] http://iccsextociclo.tripod.com/id13.html

Von Neuman[9]. La ROM interna del MCS 8051 y el MCS 8052 no pueden ser programados por el usuario. El usuario debe suministrar el programa al fabricante, y el fabricante programa los microcontroladores durante la producción. Debido a costos, la opción de la ROM programado por el fabricante no es económica para producción de pequeñas cantidades.

En la Figura 1.10 se observa la distribución de pines de un microcontrolador 8051, 8031, 8751.

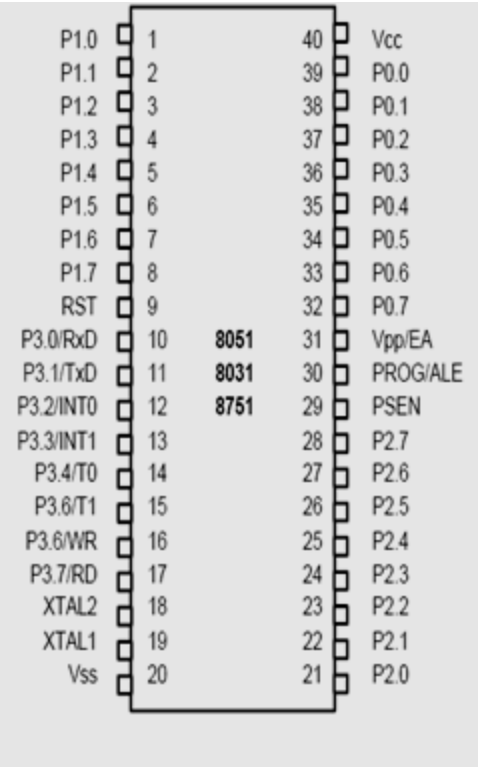


Figura 1. 10 Distribución de pines de un microcontrolador.

- El MCS 8031 (la versión sin ROM interna del MCS 8051, siendo ésta la única diferencia) tiene 128 bytes de RAM interna, posee dos temporizadores/contadores, un puerto serie, cuatro puertos de entrada/salida paralelas de propósito general (P0, P1, P2 y P3) y una lógica de control de interrupción con cinco fuentes de interrupciones.

Al lado de la RAM interna, el MCS 8031 tiene varios Registros de Funciones Especiales (SFR) (Special Function Registers) que son para control y registros de datos. Los SFRs

[9] http://www.monografias.com/trabajos12/microco/microco.shtml

también incluyen el registro acumulador A, al registro B, y el registro de estado de programa PSW (Program Status Word), que contienen las banderas del CPU.

- El MCS 8751 y el MCS 8752 son las versiones del MCS 8051 y el MCS 8052 respectivamente; con memoria EPROM (Erasable Programmable Read Only Memory); éstos pueden ser programados por los usuarios.

1.5.3 Microcontrolador SIEMENS[10]

El Siemens SAB80C515 es un miembro mejorado de la familia MCS 8051 de microcontroladores. El SAB80C515 es de tecnología CMOS que típicamente reduce los requerimientos de energía comparado a los dispositivos no-CMOS. Las características que tiene frente al MCS 8051 son más puertos, un versátil convertidor análogo a digital, un optimizado Timer 2, un watchdog timer, y modos de ahorro de energía sofisticados.

El SAB80C515 es completamente compatible con el MCS 8051; esto es, usa el mismo conjunto de instrucciones del lenguaje ensamblador MCS-51. Las nuevas facilidades del chip son controladas y monitoreadas a través de SFRs adicionales. El SAB80C515 tiene todas las SFRs del MCS 8051, y de este modo puede correr cualquier programa escrito para el MCS 8051, con excepción del uso del registro prioridad de interrupción IP. Por tanto, si un programa MCS 8051 usa prioridades de interrupción, debe ser modificado antes de que se ejecute sobre el SAB80C515.

El desinterés de modificar código MCS 8051 existente es fácilmente justificado por la disponibilidad de más fuentes de interrupción y prioridades del SAB80C515.

1.5.4 Microcontrolador MOTOROLA[11].

El 68HC11 de la familia Motorola, es un potente microcontrolador de 8 bits en su bus de datos, 16 bits en el bus de direcciones, con un conjunto de instrucciones que es similar a los más antiguos miembros de la familia 68xx (6801, 6805, 6809). Dependiendo del modelo, el 68HC11 tiene internamente los siguientes dispositivos: Memorias EEPROM o OTPROM, RAM, entradas/salidas, timers, conversores A/D, generador PWM, y canales de

[10] http://www.datasheetcatalog.net/es/datasheets_pdf/S/A/B/8/SAB80C515-M.shtml
[11] http://es.wikipedia.org/wiki/Motorola_6800

comunicación sincrónica y asincrónica (RS232 y SPI). La corriente típica que maneja es menor que 10mA.

El CPU tiene 2 acumuladores de 8 bits (A y B) que pueden ser concatenado para suministrar un acumulador doble de 16 bits (D). Dos registros índices de 16 bits están presentes (X, Y) para suministrar indexamiento para cualquier lugar dentro del mapa de memoria. El tener dos registros índices, significa que el 68HC11 es muy bueno para el procesamiento de datos. Aunque es un microcontrolador de 8 bits, el 68HC11 tiene algunas instrucciones de 16 bits como: add, subtract, divide, multiply, shift, y rotates. Un puntero de pila de 16 bits está también presente, y las instrucciones son suministradas para manipulación de la pila. Típicamente el bus de datos y direcciones están multiplexados. El temporizador comprende de un único contador de 16 bits. Viene con un convertidor A-D que es típicamente de 8 canales y 8 bits de resolución. Tiene una Interface de comunicaciones serie (SCI) - comunicaciones serie asíncrona y una Interface periférico serie (SPI) - comunicaciones serie sincrónica; el formato de datos es: 1 bit de inicio, 8 o 9 bits de datos, y un bit de parada. La velocidad puede ser desde 150 hasta 312500 Baudios (312500 para un reloj de 4MHz).

1.5.5 Microcontrolador MICROCHIP[12].

Los microcontroladores PIC de Microchip Technolohy Inc. combinan una alta calidad, bajo coste y excelente rendimiento. Un gran número de estos microcontroladores son usados en una gran cantidad de aplicaciones tan comunes como periféricos del ordenador, datos de entrada, procesamiento de datos, sistemas de seguridad y aplicaciones en el sector de telecomunicaciones.

Tanto la familia del PIC16XX, PIC17XX como PIC18FXXX, la serie que será utilizada en el desarrollo del proyecto es 18F4620, están apoyadas de sistemas de desarrollo amistosos incluido programadores, emuladores y tablas de demostración.
Asimismo estas familias están apoyadas por una gran cantidad de software incluyendo ensambladores, simuladores, etc.

[12] http://www.microchip.com

1.6 DIFERENCIAS ENTRE AUTÓMATA - MICROCONTROLADOR.

En la Tabla 1.1 se presenta un resumen de las diferencias más importantes entre un autómata y un microcontrolador.

Microcontrolador	Autómata Programable
La mayoría de los microcontroladores pueden ejecutar entre 1 millón y 5 millones de instrucciones por segundo. Es ideal para procesamientos muy rápidos en tiempo real.	Un autómata programable ejecuta entre 300 y 30.000 instrucciones por segundo.
Se programa con un lenguaje de bajo nivel tipo ensamblador o C; o bien alto nivel como el BASCOM LT, donde el usuario debe desarrollar manejadores para los diferentes periféricos.	Se programa siempre mediante un lenguaje de alto nivel, el BASIC, en el que todos los manejadores de periféricos ya están preprogramados. Con él se pueden desarrollar aplicaciones en menor tiempo.
Requiere de un periodo de aprendizaje relativamente largo, es normalmente aconsejado a técnicos experimentados.	Una persona no experimentada, puede realizar pequeños programas. Apenas requiere de tiempo de aprendizaje o conocimientos previos.
Es mucho más flexible, en el sentido de que todas las entradas/salidas son de propósito general. Esta flexibilidad obliga al usuario a desarrollar su propia electrónica.	Cada entrada/salida tiene una función preasignada, esto es, LCD, teclado, relés, triacs, optoacopladores. Toda la electrónica básica ya está desarrollada.
Su electrónica en mucho más simple, por lo que su consumo es inferior a 50 mA. Se pueden lograr aplicaciones concretas con consumos muy bajos del orden de los 5 mA.	Su consumo, es relativamente alto, no siendo adecuado para aplicaciones portátiles o de larga duración que deban alimentarse a pilas. Su consumo es superior a los 500 mA, pudiendo un sistema con 64 entradas/salidas llegar a los 3A.

Para guardar los programas residentes requiere de un programador de EPROMs externo.	No requiere de ningún dispositivo de grabación externo, todos los datos son almacenados permanentemente a través del puerto serie.
Con un mismo número de entradas/salidas tiene un precio notablemente inferior, pero debe tenerse en cuenta que trabaja con tensiones de 5 voltios y corrientes muy pequeñas del orden de miliamperios. Sus señales están disponibles en conectores de 1,27 milímetros.	Su precio es superior, pero las entradas/salidas pueden aceptar directamente tensiones de 5, 12, 24, 110 y 220 voltios. Las señales están disponibles en regletas de paso 5 milímetros.

Tabla 1. 1 Diferencias entre Autómata y Microcontrolador.

1.7 ARQUITECTURA BÁSICA.

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann, actualmente se impone la arquitectura Hardware. La arquitectura de Von Neumann 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). En la Figura 1.11 se observa la arquitectura interna de un microcontrolador.

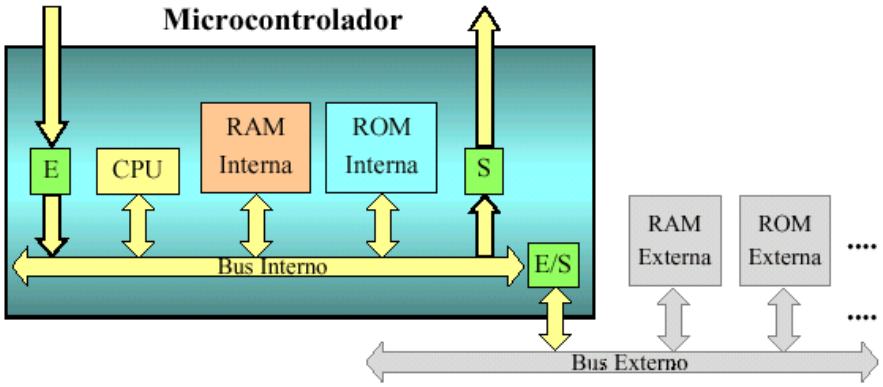


Figura 1. 11 Arquitectura interna del microcontrolador.

1.7.1 La Arquitectura Hardware.

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 escritura) simultáneamente en ambas memorias.
En la Figura 1.12 se presenta la arquitectura Hardware de un microcontrolador.

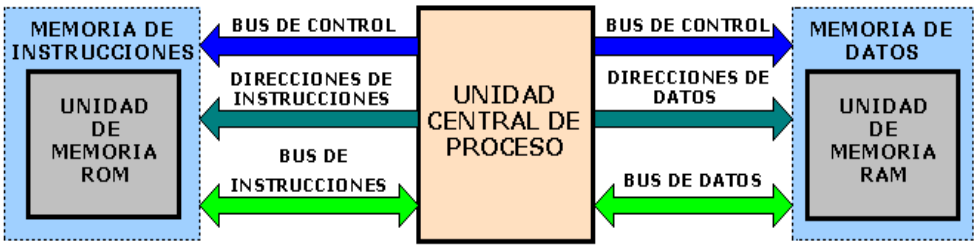


Figura 1. 12 Arquitectura hardware.

Los microcontroladores PIC responden a la arquitectura Hardware.

1.7.2 El procesador o CPU.

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software. Se encarga de direccionar la memoria de instrucciones, recibir el código de la instrucción en 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.
Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales.

- **CISC:** Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores con Set de Instrucciones Complejo). Disponen de más de 80 instrucciones máquina en su estructura, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución.
Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros.

- **RISC:** Tanto la industria de los computadores comerciales como la de los microcontroladores, se están encaminando hacia la filosofía RISC (Computadores con Set de Instrucciones Reducido). En estos procesadores el conjunto de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo.
La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

- **SISC:** En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es "específico", o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores con Set de Instrucciones Específico).

1.7.3 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 que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos.
Hay dos peculiaridades que diferencian a los microcontroladores de los computadores personales:

- No existen sistemas de almacenamiento masivo como disco duro o disquetes.
- Como el microcontrolador sólo se destina a una tarea en la memoria ROM, sólo hay que almacenar un único programa de trabajo.

La RAM en estos dispositivos es de poca capacidad, pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM, pues se ejecuta directamente desde la ROM.

Los usuarios de computadores personales están habituados a manejar Megabytes de memoria, pero, los diseñadores con microcontroladores trabajan con capacidades de ROM comprendidas entre 512 bytes a 8 k bytes de RAM.

Según el tipo de memoria ROM que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Se describen las cinco versiones de memoria no volátil que se pueden encontrar en los microcontroladores del mercado.

1.7.3.1 ROM con Máscara.

Es una memoria no volátil de sólo lectura, cuyo contenido se graba durante la fabricación del chip. El elevado coste del diseño de la máscara, sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria, cuando se precisan cantidades superiores a varios miles de unidades.

1.7.3.2 OTP.

El microcontrolador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario. OTP (One Time Programmable). Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC.

La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas.
Tanto en este tipo de memoria como en la EPROM, se suele usar la encapsulación mediante fusibles para proteger el código contenido.

1.7.3.3 EPROM.

Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read OnIy Memory) pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.

1.7.3.4 EEPROM.

Se trata de memorias de sólo lectura, programables y borrables eléctricamente EEPROM (Electrical Erasable Programmable Read OnIy Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y borrado. No disponen de ventana de cristal en la superficie.

Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.
El número de veces que puede grabarse y borrarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación continua. Son muy idóneos para la enseñanza y la Ingeniería de diseño.

Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que disponen dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

1.7.3.5 FLASH.

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos potencia y es más pequeña.
A diferencia de la ROM, la memoria FLASH es programable en el circuito; es más rápida y de mayor densidad que la EEPROM.

La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil, es más veloz y tolera más ciclos de escritura/borrado.

Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados "en circuito"; es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado,

por ejemplo el control del motor de un automóvil, permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores tales como la compresión, la instalación de nuevas piezas, etc. La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

1.8 RECURSOS FUNDAMENTALES.

1.8.1 Puertos de Entrada y Salida.

La principal utilidad de los pines contiene un microcontrolador es soportar las líneas de E/S que comunican al CPU interno con los periféricos exteriores.
Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

1.8.2 Reloj Principal.

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema.

Generalmente, el circuito oscilador 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.

1.8.3 Recursos Especiales.

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador. En algunas, se amplía las capacidades de las memorias, en otras se

incorpora nuevos recursos, en otras se reduce las prestaciones al mínimo para aplicaciones muy simples, etc.

Los principales recursos específicos que incorporan los microcontroladores son:
• Temporizadores o "Timers".
• Perro guardián o "Watchdog".
• Protección ante fallo de alimentación o "Brownout".
• Estado de reposo o de bajo consumo.
• Conversor A/D.
• Conversor D/A.
• Comparador analógico.
• Modulador de anchura de pulsos o PWM.
• Puertos de E/S digitales.
• Puertos de comunicación.

- **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 los pines del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

- **Perro guardián o "Watchdog".**

El "Watch Dog Timer" o "Temporizador Perro guardián" es un artificio que se utiliza para evitar que los microprocesadores se queden "colgados".
No es extraño que en electrónica se den circunstancias, no previstas por el diseñador en las que un microprocesador se quede en un estado indeterminado del que le sea imposible salir sin una ayuda externa.

El Watchdog lo que hace fundamentalmente es resetear el microcontrolador tras un periodo de tiempo determinado. Su funcionamiento es similar a la Interrupción por Desbordamiento de un Timer, que se produce cuando un Timer que es incrementado continuamente pasa de su valor máximo al mínimo para comenzar de nuevo a contar. En el caso del Watchdog, en lugar de saltar una interrupción, se genera un reset automático en el momento de producirse dicho desbordamiento.

- **Protección ante fallo de alimentación o "Brownout".**

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (Vcc) 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.

- **Estado de reposo o de bajo consumo.**

Son abundantes las situaciones reales de trabajo en que el microcontrolador no debe trabajar, debe esperar 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 SLEEP en los PIC, que permiten pasar 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 (ADC).**

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 ADC diversas señales analógicas desde los pines del circuito integrado microcontrolador.

- **Conversor D/A (DAC)**

Transforma los datos digitales obtenidos del procesamiento del CPU en su correspondiente señal analógica que envía al exterior por uno de los pines del microcontrolador. Existen muchos conversores 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 uno de los pines del encapsulado. 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 pulsos o PWM.**

Son circuitos que proporcionan en su salida pulsos de ancho variable, que se ofrecen al exterior a través de los pines del encapsulado. Esta señal puede servir por ejemplo, para controlar la velocidad de motores DC, previo de pasar la señal por un circuito de acondicionamiento.

- **Puertos de E/S digitales.**

Todos los microcontroladores destinan algunos de sus pines para soportar líneas de E/S digitales. 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, cargando un 1 ó un 0, en el bit correspondiente de un registro destinado a su configuración.

- **Puertos de comunicación.**

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, como buses de microprocesadores, buses de sistemas, buses de

redes, o poder adaptarlos con otros elementos bajo otras normas y protocolos, algunos modelos disponen de recursos que permiten realizar 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), que es un moderno bus serie para los PC.
Bus I2C, que es un interfaz serie de dos hilos desarrollado por Philips.
CAN (Controller Area Network), para permitir la adaptación con redes de conexionado multiplexado, desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles.

1.9 HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES.

Uno de los factores que más importancia a la hora de seleccionar un microcontrolador, entre otros; es el soporte tanto de software como de hardware que dispone. Un buen conjunto de herramientas de desarrollo puede ser decisivo en la elección, ya que pueden suponer una ayuda inestimable en el desarrollo del proyecto.
Las principales herramientas de ayuda al desarrollo de sistemas basados en microcontroladores se indican a continuación:

1.9.1 Desarrollo del Software.

- **Ensamblador**. La programación en lenguaje ensamblador puede resultar un tanto ardua para el principiante, pero permite desarrollar programas muy eficientes, ya que otorga al programador el dominio absoluto del sistema. Los fabricantes suelen proporcionar el programa ensamblador de forma gratuita; en cualquier caso, siempre se puede encontrar una versión gratuita para los microcontroladores más populares.

- **Compilador**. La programación en un lenguaje de alto nivel (como el C ó el Basic) permite disminuir el tiempo de desarrollo de una aplicación. No obstante, si no se

programa con cuidado, el código resultante puede ser mucho más ineficiente que el programado en ensamblador. Las versiones más potentes suelen ser muy caras, aunque para los microcontroladores más populares pueden encontrarse versiones demo, limitadas e incluso compiladores gratuitos.

- **Depuración.** Debido a que los microcontroladores van a controlar dispositivos físicos, los desarrolladores necesitan herramientas que les permitan comprobar el buen funcionamiento del microcontrolador cuando es conectado al resto de circuitos, los cuales se realiza a través de la depuración de los programas de aplicación.

- **Simulador.** Son capaces 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 se ahorrará 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. El sistema operativo de la placa 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.

1.10 COMUNICACIONES.

1.10.1 RS-232C.

El puerto serie RS-232C, presente en todos los ordenadores actuales, es la forma más comúnmente usada para realizar transmisiones de datos entre ordenadores. El RS-232C es un estándar que constituye la tercera revisión de la antigua norma RS-232, propuesta por la

EIA (Asociación de Industrias Electrónicas), realizándose posteriormente una versión internacional por el CCITT[13], conocida como V.24. Las diferencias entre ambas son mínimas, por lo que a veces se habla indistintamente de V.24 y de RS-232C (incluso sin el sufijo "C"), refiriéndose siempre al mismo estándar.

El RS-232C consiste en un conector tipo DB-25 de 25 pines, aunque es normal encontrar la versión de 9 pines DB-9, más barato e incluso más extendido para cierto tipo de periféricos (como el ratón serie del PC). En cualquier caso, los PCs no suelen emplear más de 9 pines en el conector DB-25.

Las señales con las que trabaja este puerto serie son digitales, de +12V (0 lógico) y -12V (1 lógico), para la entrada y salida de datos, y a la inversa en las señales de control. El estado de reposo, en la entrada y salida de datos es -12V.

Dependiendo de la velocidad de transmisión empleada, es posible tener cables de hasta 15 metros.

En la Tabla 1.2 se presenta la descripción de pines de puerto serie RS-232C.

Pin	**Función**
TXD	(Transmitir Datos)
RXD	(Recibir Datos)
DTR	(Terminal de Datos Listo)
DSR	(Equipo de Datos Listo)
RTS	(Solicitud de Envío)
CTS	(Libre para Envío)
DCD	(Detección de Portadora)

Tabla 1. 2 Descripción de pines del puerto RS 232C.

Las señales TXD, DTR y RTS son de salida, mientras que RXD, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es SG (Tierra de Señal).

[13] **Comité Consultivo Internacional Telegráfico y Telefónico** (*Consultative Committee for International Telegraphy and Telephony*.

En la Tabla 1.3 se presenta la descripción de pines del puerto serial, para los conectores DB-25 y DB-9.

Número de Pin		Señal	Descripción	E/S
En DB-25	En DB-9			
1	1	-	Masa chasis	-
2	3	TxD	Transmit Data	S
3	2	RxD	Receive Data	E
4	7	RTS	Request To Send	S
5	8	CTS	Clear To Send	E
6	6	DSR	Data Set Ready	E
7	5	SG	Signal Ground	-
8	1	CD/DCD	(Data) Carrier Detect	E
15	-	TxC(*)	Transmit Clock	S
17	-	RxC(*)	Receive Clock	E
20	4	DTR	Data Terminal Ready	S
22	9	RI	Ring Indicator	E
24	-	RTxC(*)	Transmit/Receive Clock	S

Tabla 1. 3 Señales de entradas y salidas del puerto serie.

A continuación en la Tabla 1.4 se observa la estructura física de los conectores DB-25 y DB-9.

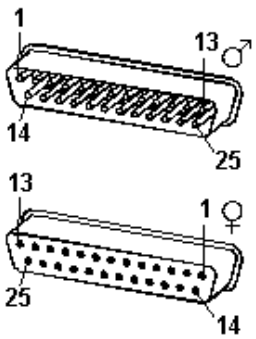
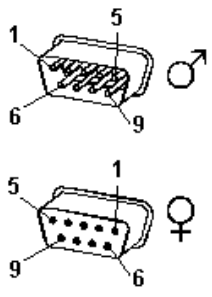
Conector DB 25	**Conector DB 9**
	

Tabla 1. 4 Conector DB 25 y conector DB9.

El término "serial" viene del hecho de que el puerto serie "serializa" los datos. Esto quiere decir que toma un byte de datos y transmite los 8 bits del byte uno a la vez. La ventaja del puerto serie es que necesita únicamente 1 solo cable para transmitir los 8 bits (mientras que un puerto paralelo necesita 8). La desventaja es que dura 8 veces más para transmitir el dato que si tuviera 8 cables.

Antes de cada byte de información, el puerto serial manda un bit de comienzo (start bit), el cual es un bit con valor de 0. Después de cada byte de datos, este manda un bit de parada (stop bit) para indicar que el byte ha sido completado. Algunas veces también se manda un bit de paridad. Los puertos serie, también llamados **puertos de comunicación (COM)**, son **bi-direccionales**. La comunicación bidireccional permite a cada dispositivo recibir datos, así como también transmitirlos. Los dispositivos seriales usan distintos pines para recibir y transmitir datos. Usando el mismo pin, limitaría la comunicación a **half-duplex**, esto quiere decir que la información solamente podría viajar en una dirección a la vez. Usando distintos pines, permite que la comunicación sea **full-duplex**, en la cual la información puede viajar en ambas direcciones al mismo tiempo.

Mediante los puertos de E/S se pueden intercambiar datos, mientras que las IRQ producen una interrupción para indicar a la CPU que ha ocurrido un evento (por ejemplo, que ha llegado un dato, o que ha cambiado el estado de algunas señales de entrada). La CPU debe responder a estas interrupciones lo más rápido posible, para que de tiempo a recoger el dato antes de que el siguiente lo sobrescriba. Sin embargo, las UART 16550A incluyen unos buffers de tipo FIFO, dos de 16 bytes (para recepción y transmisión); donde se pueden guardar varios datos, antes de que la CPU los recoja.

Esto también disminuye el número de interrupciones por segundo generadas por el puerto serie.

Siempre que se va establecer comunicación a través de un puerto serial, es necesario definir algunos parámetros de tal forma que coincidan en ambos equipos. Estos parámetros son: la velocidad, la paridad, los bits de parada y el control de flujo.

- **Velocidad (baud rate).** Dada en baudios (cambio de nivel de voltaje por segundo), indica la transmisión de información que se usará entre los equipos.

- **Paridad (parity).** Es un bit que indica si la suma de los bits de datos es par o impar. Se utiliza como mensaje para verificar si la información recibida es la correcta.

- **Bits de datos (data bits).** Cada dato o carácter puede estar representado a través de 7 u 8 bits en forma serial.

- **Bits de parada (stop bits).** Son los bits utilizados para separar un dato del siguiente.

- **Control de Flujo (handshake).** Sirve para indicar la forma en que cada equipo controlará el envío y el recibido de información. El control handshake utiliza líneas adicionales, mientras que el control por software se hace por las mismas de transmisión de datos.

Para establecer comunicación serial entre dos equipos, los parámetros que se describen anteriormente deben ser exactamente iguales.

1.10.2 USB[14].

El Bus de Serie Universal (USB, de sus siglas en inglés Universal Serial Bus) es una interfaz que provee un estándar de bus serie para conectar dispositivos a un ordenador personal (generalmente a un PC). Un sistema USB tiene un diseño asimétrico, que consiste en un solo servidor y múltiples dispositivos conectados en una estructura de árbol utilizando concentradores especiales. Se pueden conectar hasta 127 dispositivos a un sólo servidor, pero la suma debe incluir a los concentradores también, así que el total de dispositivos realmente usables es algo menor.

Fue creado en 1996 por siete empresas: IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC.
El estándar incluye la transmisión de energía eléctrica al dispositivo conectado. Algunos dispositivos requieren una potencia mínima, así que se pueden conectar varios sin necesitar fuentes de alimentación extra. La mayoría de los concentradores incluyen fuentes de alimentación que brindan energía a los dispositivos conectados a ellos, pero algunos

[14] http://mx.geocities.com/pcmuseo/mecatronica/usb.htm

dispositivos consumen tanta energía que necesitan su propia fuente de alimentación. Los concentradores con fuente de alimentación pueden proporcionarle corriente eléctrica a otros dispositivos sin quitarle corriente al resto de la conexión (dentro de ciertos límites).
El diseño del USB tenía en mente eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos bus ISA o PCI, y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Cuando se conecta un nuevo dispositivo, el servidor lo enumera y agrega el software necesario para que pueda funcionar.

El USB puede conectar periféricos como ratones, teclados, escáneres, cámaras digitales, impresoras, discos duros, tarjetas de sonido y componentes de red. Para dispositivos multimedia como escáneres y cámaras digitales, el USB se ha convertido en el método estándar de conexión. Para impresoras, el USB ha crecido tanto en popularidad que ha empezado a desplazar a los puertos paralelos porque el USB hace sencillo el poder agregar más de una impresora a un ordenador personal.

En el caso de los discos duros, el USB es poco probable que reemplace completamente a los buses como el ATA (IDE) y el SCSI porque el USB tiene un rendimiento un poco más lento que esos otros estándares. El nuevo estándar Serial ATA permite tasas de transferencia de hasta aproximadamente 150 MB por segundo. Sin embargo, el USB tiene una importante ventaja en su habilidad de poder instalar y desinstalar dispositivos sin tener que abrir el sistema, lo cual es útil para dispositivos de almacenamiento desinstalables. Hoy en día, una gran parte de los fabricantes ofrece dispositivos USB portátiles que ofrecen un rendimiento casi indistinguible en comparación con los ATA (IDE).
En la Figura 1.13 se puede observar la tarjeta para USB.

Figura 1. 13 Tarjeta PCI-USB 2.0.

El estándar USB 1.1 tenía dos velocidades de transferencia: 1.5 Mbit/s para teclados, ratón, joysticks, etc., y velocidad completa a 12 Mbit/s. La mayor ventaja del estándar USB 2.0 es añadir un modo de alta velocidad de 480 Mbit/s. En su velocidad más alta, el USB compite directamente con FireWire (excepto en el área de cámaras digitales portátiles, el USB tiene limitaciones tecnológicas que prohíben su uso viable en esta área).

Las especificaciones USB 1.0, 1.1 y 2.0 definen dos tipos de conectores para conectar dispositivos al servidor: A y B. Sin embargo, la capa mecánica ha cambiado en algunos conectores. Por ejemplo, el IBM UltraPort es un conector USB privado localizado en la parte superior del LCD de los ordenadores portátiles de IBM. Utiliza un conector mecánico diferente mientras mantiene las señales y protocolos característicos del USB. Otros fabricantes de artículos pequeños han desarrollado también sus medios de conexión pequeños, y una gran variedad de ellos ha aparecido. Algunos de baja calidad.
Una extensión del USB llamada "USB-On-The-Go" permite a un puerto actuar como servidor o como dispositivo - esto se determina por qué lado del cable está conectado al aparato.

En la Figura 1.14 se observa dos tipos de conexiones, llamados tipo A y tipo B.

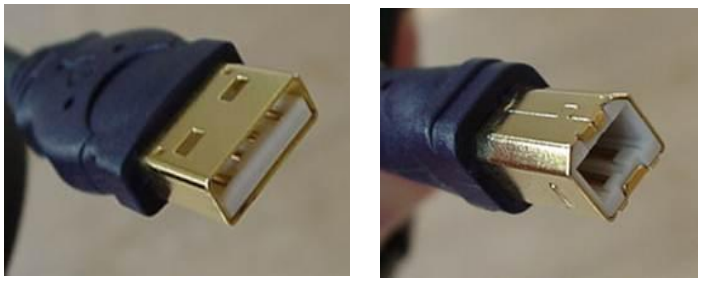


Figura 1. 14 Conector típico de USB, llamado conexión "A", y conexión tipo "B".

Incluso después que el cable está conectado y las unidades se están comunicando, las 2 unidades pueden "cambiar de papel" bajo el control de un programa. Esta facilidad está específicamente diseñada para dispositivos como PDA, donde el enlace USB podría conectarse a un PC como un dispositivo, y conectarse como servidor a un teclado o ratón. El "USB-On-The-Go" también ha diseñado 2 conectores pequeños, el mini-A y el mini-B, así que esto debería detener la proliferación de conectores miniaturizados de entrada.

1.10.4 Periféricos.

El USB soporta periféricos de baja y media velocidad. Empleando dos velocidades para la transmisión de datos, de 1.5 y 12 Mbps, se consigue una utilización más eficiente de sus recursos. Los periféricos de baja velocidad tales como teclados, ratones, joysticks, y otros periféricos para juegos, no requieren 12 Mbps. Empleando para ellos 1,5 Mbps, se puede dedicar más recursos del sistema a periféricos tales como monitores, impresoras, módems, scanner, equipos de audio, que precisan de velocidades más altas para transmitir mayor volumen de datos o datos cuya dependencia temporal es más estricta .

En las Figuras 1.15 y 1.16, se puede ver cómo los hubs proporcionan conectividad a toda una serie de dispositivos periféricos.

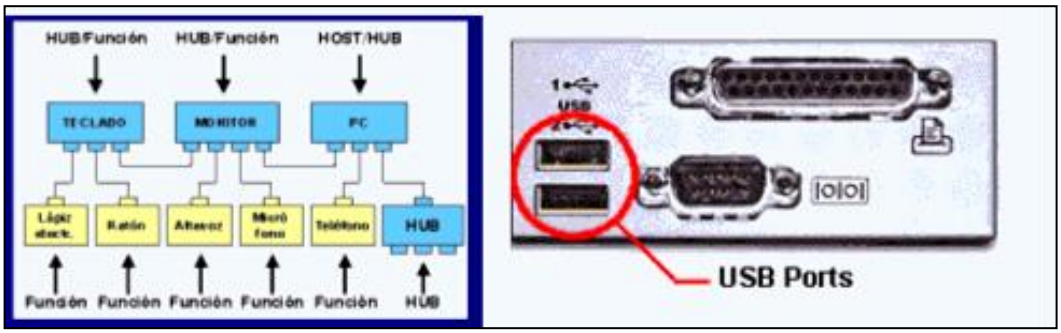


Figura 1. 15 Conectividad entre bus USB.

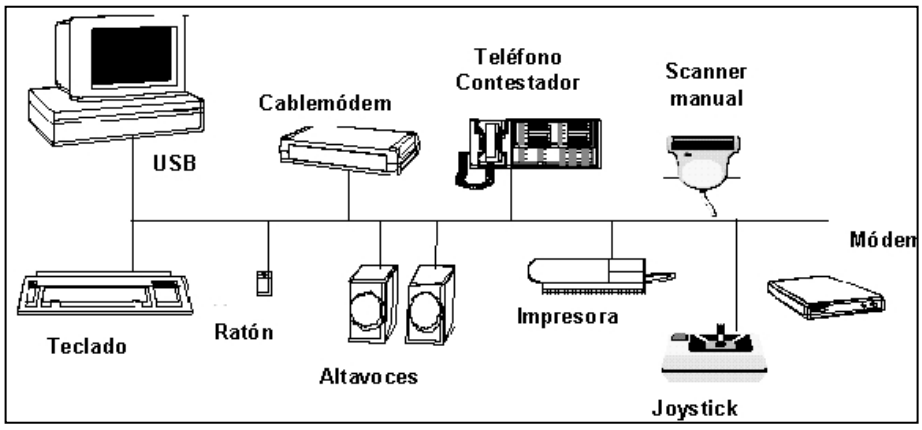


Figura 1. 16 Dispositivos USB conectados a un PC.

En la Figura 1.17, se observa un dispositivo Hub de 4 vías.

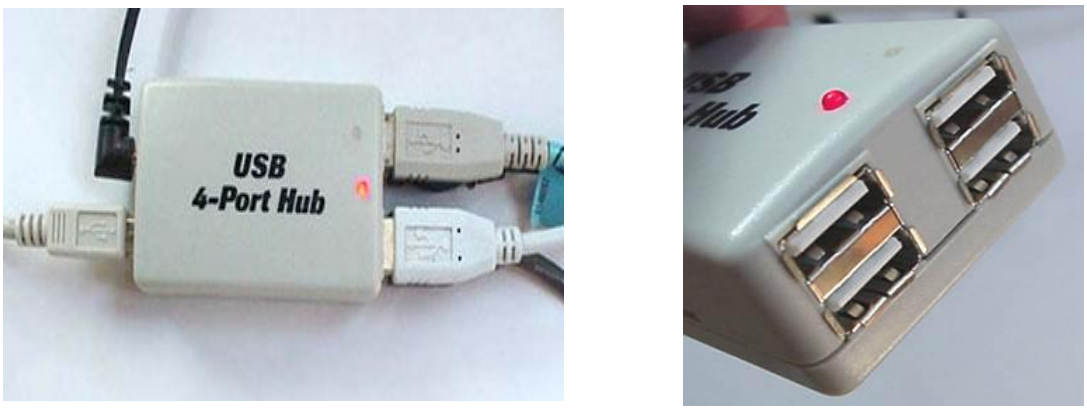


Figura 1. 17 Hub de 4 entradas USB tipo "A".

1.10.5 Cables y Conectores.

USB transfiere señales y energía a los periféricos utilizando un cable de 4 hilos, apantallado para transmisiones a 12 Mbps y no apantallado para transmisiones a 1.5 Mbps. En la Figura 1.18, se muestra un esquema del cable, con dos conductores para alimentación y los otros dos para señal, debiendo estos últimos ser trenzados o no, según la velocidad de transmisión.

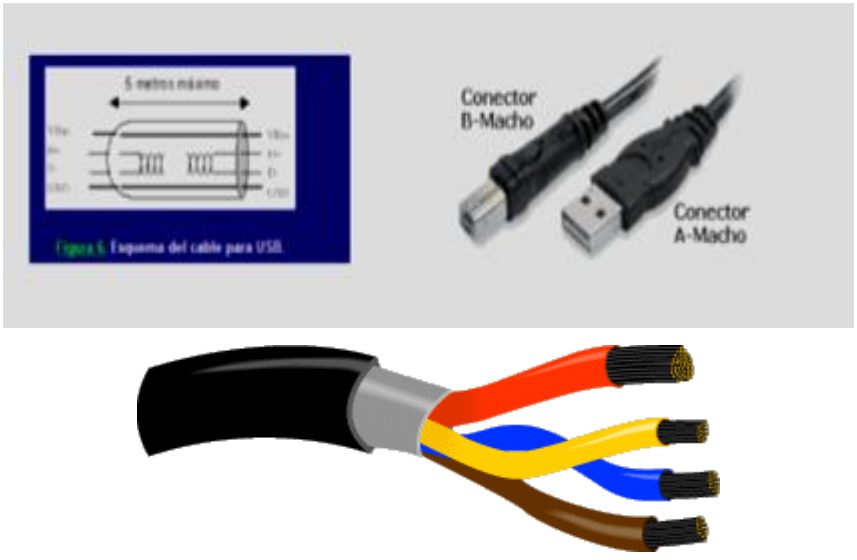


Figura 1. 18 Esquema del cable, con dos conductores para alimentación y señal.

Dentro de un cable de USB existen 2 cables para alimentación: +5 volts (rojo) y tierra (café) y un par de cables trenzados (amarillo y azul) para transportar los datos.

El calibre de los conductores destinados a alimentación de los periféricos varía desde 20 a 26 AWG, mientras que el de los conductores de señal es de 28 AWG . La longitud máxima de los cables es de 5 metros.

En lo que respecta a los conectores, éstos son del tipo ficha (o conector) y receptáculo, y son de dos tipos: serie A y serie B. Los primeros presentan las cuatro patillas correspondientes a los cuatro conductores alineadas en un plano. El color recomendado es blanco sucio y los receptáculos se presentan en cuatro variantes: vertical, en ángulo recto, panel y apilado en ángulo recto, así como para montaje pasamuro. Se emplean en aquellos dispositivos en los que el cable externo, está permanentemente unido a los mismos, tales como teclados, ratones, y hubs o concentradores.
Los conectores de la serie B presentan los contactos distribuidos en dos planos paralelos, dos en cada plano, y se emplean en los dispositivos que deban tener un receptáculo al que se pueda conectar un cable USB. Por ejemplo impresoras, scanner, y módems.

1.10.6 Radio Frecuencia[15].

En la actualidad, la tecnología más extendida para la identificación de objetos es la de los códigos de barras. Sin embargo, éstos presentan algunas desventajas, como son la escasa cantidad de datos que pueden almacenar y la imposibilidad de ser modificados (reprogramados). Una mejora constituye el origen de la tecnología RFID, consiste en utilizar chips de silicio que pudieran transferir los datos que almacenaban al lector sin contacto físico (de forma equivalente a los lectores de infrarrojos utilizados para leer los códigos de barras).

La **RFID (**siglas de **R**adio **F**requency **ID**entification, en español *Identificación por radiofrecuencia*) es una tecnología en crecimiento que utiliza frecuencias de radio para reconocer productos mediante un pequeño chip que puede estar insertado en una etiqueta (etiqueta RFID), lo que permite realizar un rastreo de su localización. La distancia de rastreo varía mucho, dependiendo del tamaño, tipo y antena del chip, pero podría ser desde

[15] http://es.wikipedia.org/wiki/RFID.

un par de cm. a varios kilómetros en los más complejos, lo que evidentemente incorpora un grado de comodidad en su utilización. La etiqueta RFID posee un número de identificación único que lo diferencia del código de barras el cual es el mismo para un grupo de productos.

Un sitema RFID consta de los siguientes componentes:

- **Etiqueta RFID** o **transpondedor**: compuesta por una antena, un transductor radio y un material encapsulado o chip. El propósito de la antena es permitirle al chip, el cual contiene la información, transmitir la información de identificación de la etiqueta. Existen varios tipos de etiquetas. El chip posee una memoria interna con una capacidad que depende del modelo y varía de una decena a millares de bytes.
- **Lector de RFID** o **transceptor**: compuesto por una antena, un transceptor y un decodificador. El lector envía periódicamente señales para ver si hay alguna etiqueta en sus inmediaciones. Cuando capta una señal de una etiqueta (la cual contiene la información de identificación de ésta), extrae la información y se la pasa al subsistema de procesamiento de datos.
- **Subsistema de procesamiento de datos**: proporciona los medios de proceso y almacenamiento de datos.

En la Figura 1.19 se puede observar los módulos de radio frecuencia.

Figura 1. 19 Series TLP/A (Tx) y RLP/A(Rx).

Es un sistema de almacenamiento y recuperación de datos remotos, que usa dispositivos denominados etiquetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas **Auto ID** (*Automatic Identification*, o Identificación Automática).

Las regulaciones de frecuencia son en las etiquetas RFID de baja frecuencia (LF: 125 - 134 kHz y 140 - 148.5 kHz) y de alta frecuencia (HF: 13.56 MHz) se pueden utilizar de forma global sin necesidad de licencia. La frecuencia ultra alta (UHF: 868 - 928 MHz), no puede ser utilizada de forma global, ya que no hay un único estándar global. En Norteamérica, la frecuencia ultra elevada se puede utilizar sin licencia para frecuencias entre 908 - 928 MHz, pero hay restricciones en la energía de transmisión. En Europa la frecuencia ultra elevada está bajo consideración para 865.6 - 867.6 MHz. Su uso es sin licencia sólo para el rango de 869.40 - 869.65 MHz, pero existen restricciones en la energía de transmisión. El estándar UHF norteamericano (908-928 MHz) no es aceptado en Francia e Italia, ya que interfiere con sus bandas militares. En China y Japón no hay regulación para el uso de la frecuencia ultra elevada. Cada aplicación de frecuencia ultra elevada en estos países necesita de una licencia, que debe ser solicitada a las autoridades locales, y puede ser revocada. En Australia y Nueva Zelanda, el rango es de 918 - 926 MHz para uso sin licencia, pero hay restricciones en la energía de transmisión.

1.10.6.1 Áreas de Aplicación.

Dependiendo de las frecuencias utilizadas en los sistemas RFID, el coste, el alcance y las aplicaciones son diferentes. Los sistemas que emplean frecuencias bajas tienen igualmente costes bajos, pero también baja distancia de uso. Los que emplean frecuencias más altas proporcionan distancias mayores de lectura y velocidades de lectura más rápidas. Así, las de baja frecuencia se utilizan comúnmente para la identificación de animales, seguimiento de barricas de cerveza, o como llave de automóviles con sistema antirrobo. En ocasiones se insertan en pequeños chips en mascotas, para que puedan ser devueltas a su dueño en caso de pérdida. En los Estados Unidos se utilizan dos frecuencias para RFID: 125 kHz (el estándar original) y 134,5 kHz (el estándar internacional). Las etiquetas RFID de alta frecuencia se utilizan en bibliotecas y seguimiento de libros, control de acceso en edificios, seguimiento de equipaje en aerolíneas, seguimiento de artículos de ropa y ahora último en pacientes de centros hospitalarios para hacer un seguimiento de su historia clínica. Un uso extendido de las etiquetas de alta frecuencia como identificación de acreditaciones, substituyendo a las anteriores tarjetas de banda magnética. Sólo es necesario acercar estas insignias a un lector para autenticar al portador.

En la Figura 1. 20 se observa uno de los usos de RFID

Figura 1. 20 Etiqueta RFID utilizada para el cobro de peaje.

A largo plazo, cualquiera que tuviera un lector de chips RFID podría conocer de los aparatos, electrodomésticos, ropa, que se dispone en casa sin necesidad de entrar. Un chip RFID en el móvil, reloj o cartera podría permitir saber si una persona ha estado en un determinado establecimiento.

CAPÍTULO II

2. SELECCIÓN DEL HADWARE PARA EL PLC

2.1 SELECCIÓN DEL MICROCONTROLADOR[16]

Al seleccionar un microcontrolador para una aplicación es necesario tener en cuenta una serie de factores:

2.1.1 Costo: El precio de un producto que incorpore un microcontrolador no sólo depende del precio del microcontrolador, sino también del costo del diseño de la placa, desarrollo del software, precio de otros componentes, etc. Esto se debe tener en cuenta para el desarrollo de un proyecto, quien debe percatarse de las herramientas de desarrollo de las que dispone: simuladores, compiladores, etc. Para reducir costo, los fabricantes de productos que incorporan microcontroladores suelen ser fieles a las marcas, de forma que puedan usar las herramientas de desarrollo en distintas aplicaciones.

2.1.2 Características de la Aplicación: Antes de seleccionar el microcontrolador es necesario analizar los requisitos de la aplicación:

- **Procesamiento de datos:** En algunas aplicaciones es necesaria la realización de cálculos, por lo tanto es preciso elegir un microcontrolador suficientemente rápido. También se debe de tener en cuenta la precisión de los datos a manejar: si no es suficiente con un microcontrolador de 8 bits, acudir a arquitecturas de 16 o 32 bits.

- **Entradas/Salidas:** Se debe elegir el microcontrolador que se adapte a los requerimientos de Entradas y Salidas (E/S): tipo (analógico o digital).

[16] http://laimbio08.escet.urjc.es/assets/files/tema1.pdf

- **Consumo:** Varios equipos que incorporan un microcontrolador se alimentan con baterías. En este caso conseguir reducir el consumo es vital para la viabilidad del producto.

- **Memoria:** Para determinar la cantidad de memoria necesaria es imprescindible realizar una versión preliminar de la aplicación para estimar cuanta memoria volátil o no será necesaria y si es conveniente disponer de memoria no volátil modificable.

- **Ancho de palabra:** El criterio, teniendo en cuenta que el precio aumenta con la longitud de la palabra, debe ser seleccionar el microcontrolador con menor ancho de palabra que se adapte a los requerimientos de la aplicación.

- **Diseño de la placa:** La selección de un microcontrolador u otro condiciona el diseño de la placa. Se debe tener en cuenta que quizás usar un microcontrolador barato encarezca el resto de componentes del diseño.

2.2 EL MICROCONTROLADOR PIC18F4620.[17]

Ofrece las ventajas de todos los microcontroladores de la familia PIC18, alto rendimiento, a un precio bajo, memoria de programa mejorada. Este integrado de la familia PIC, forma parte de los circuitos integrados de Microchip Technology Inc., que pertenece a la categoría de los microcontroladores. Presenta características muy versátiles que permiten aplicaciones en un gran número de variedades.

Este microcontrolador como cualquier circuito integrado analógico tiene entradas, salidas y algunos componentes exteriores necesarios para procesar las señales de entrada y convertirlas en las señales de salida (Figura 2.1). El 18F4620 requiere un cristal con dos capacitores y como mínimo un resistor para el reset. Por supuesto necesita una tensión de fuente de 5V (VDD) aplicada con respecto al terminal de masa (VSS). Posee cuatro puertos de salida, el A, B, C y D.

El terminal 1 opera como reset, pero también cumple funciones de carga de memoria de programa cuando es excitado con pulsos de 15V. El terminal RA4 también tiene funciones como entrada de un temporizador; y RBO a RB2, cumple también funciones como entrada de interrupción.

[17] http://www.microchip.com

Las características principales que posee el PIC son las siguientes:

- Memoria Flash de 32 Kbytes.
- Número de instrucciones: 32768.
- RAM (incluida) de 3968 bytes.
- Datos EEPROM 1024 bytes.

A continuación en la Tabla 2.1, se detalla las principales características del PIC 18F4620.

Features	PIC18F2525	PIC18F2620	PIC18F4525	PIC18F4620
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	49152	65536	49152	65536
Program Memory (Instructions)	24576	32768	24576	32768
Data Memory (Bytes)	3968	3968	3968	3968
Data EEPROM Memory (Bytes)	1024	1024	1024	1024
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/Compare/ PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	No	No	Yes	Yes
10-bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), $\overline{MCLR}$ (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin SPDIP 28-pin SOIC	28-pin SPDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

Tabla 2. 1 Características del PIC 18F4620.

2.2.1 Características Periféricas.

- Posee 5 puertos I/O, de 8 bits; menos el puerto A, que sólo tiene 7 bits.
- Compatibilidad A/D de 10 bits.
- Tiene un generador de oscilación que le brindan características de timer.

A continuación en la Figura 2.1 se muestra las entradas y salidas del microcontrolador.

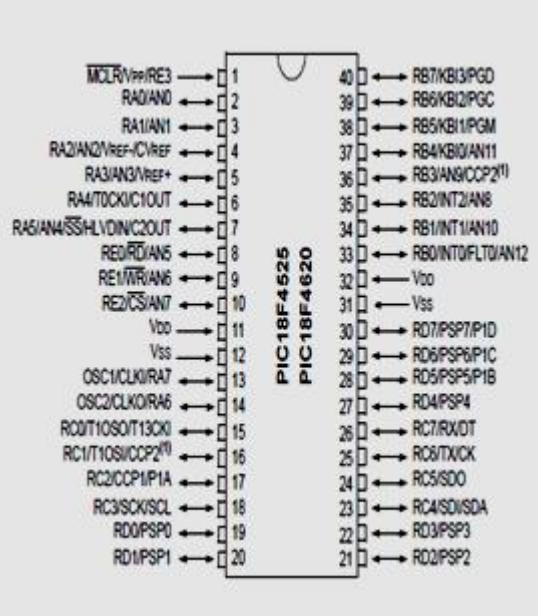


Figura 2. 1 PIC 18F4620.

2.2.2 Arquitectura Interna[18].

La mayoría de los microcontroladores se comportan de forma similar, por ello se referirá al microcontrolador PIC 18F4620, el cual es utilizado en el proyecto; cuya arquitectura interna puede observarse en la Figura 2.2.

Observe primero los bloques externos. Existe un cristal que se conecta en OSC1 y OSC2 para generar el CLOCK del sistema. Luego, una señal de entrada llamada MCLR negada, que es un nombre que se le da para el RESET (debido a que ese pin tiene un doble uso) y, por último, dos puertos paralelos de I/O (entrada o salida) llamados puerto A y puerto B.
Una de los pines del puerto A puede ser utilizada como entrada de interrupciones (este pin especial hace que el microprocesador deje de realizar la tarea que estaba ejecutando y pase a realizar otra tarea alternativa; cuando la termina, vuelve a su programa original).

[18] www.webelectronica.com.ar

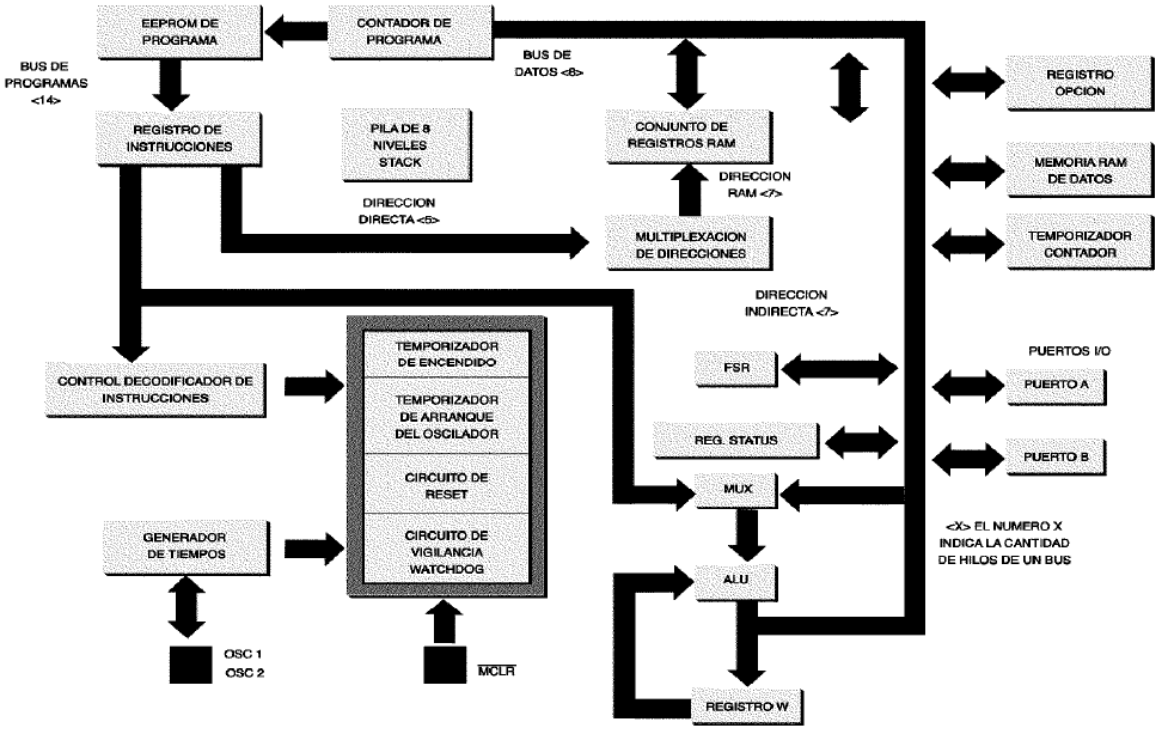


Figura 2. 2 Arquitectura interna del PIC 18F4620.

Uno de los pines del puerto A puede ser utilizado como entrada de interrupciones (este pin especial hace que el microprocesador deje de realizar la tarea que estaba ejecutando y pase a realizar otra tarea alternativa; cuando la termina vuelve a su programa original).

En el bloque más grande (temporizadores), se observa un grupo de bloques dedicados a mejorar el funcionamiento pero sin influir directamente en el flujo de señales. Se observa un temporizador de encendido, un temporizador de arranque del oscilador de CLOCK, un circuito de reset y un circuito llamado de vigilancia o WATCHDOG. Los dos primeros bloques procuran un arranque ordenado para no producir una carga al mismo tiempo sobre la fuente. Por último, existe un circuito con el nombre: "perro guardián". Su función es estar vigilante el máximo de tiempo que tarda el microprocesador en completar su programa, y en caso de superarse ese tiempo, provocar un reset automático porque el microprocesador se quedó bloqueado en alguna parte de su programa. También se dice que el microprocesador se quedó colgado o congelado.

Este bloque de circuitos no trabaja independientemente sino que requiere conexiones al exterior y al interior del dispositivo. Por ejemplo, no siempre son utilizados y es el programa quien determina su utilización y además ajusta sus parámetros. Esto se realiza a través del bloque de control o decodificador de instrucciones.

En la sección de arriba a la izquierda, se observa la memoria de programa, el contador de programa, el registro de instrucciones y la pila o STACK de 8 niveles. Cuando se habla de registros se refiere a pequeñas unidades de memoria transitoria, construida por lo general con un registro de desplazamiento. Son memorias volátiles que se utilizan para guardar información por un tiempo mínimo, con el fin de realizar una operación compleja de varios pasos.

El contador de programa es el responsable de que el microprocesador vaya analizando las instrucciones en orden ascendente. Éste guarda el número de instrucción en el STACK y la instrucción misma la pasa al registro de instrucciones desde donde se envía al resto del microprocesador.

El STACK es, una pila de registros (en el ejemplo hay 8), debido a que el programa puede tener derivaciones. Cuando se termina de ejecutar un loop se debe volver al mismo punto del programa en donde se había producido la bifurcación y eso es posible porque ese número de instrucción quedó guardado en uno de los registros de la pila. Es común que un loop tenga, a su vez, un loop secundario y cuando se ejecuta ese loop secundario se debe volver al mismo punto del loop primario, eso se consigue con guardar ese número de instrucción del loop secundario en otro registro de la pila.

En la sección inferior derecha, se ubican los bloques responsables de efectuar operaciones matemáticas y lógicas binarias; el ALU (Figura 2.3), proviene de Aritmetic Logic Unit (Unidad Aritmética y Lógica). En este sector es imprescindible utilizar un registro ya que una operación aritmética o lógica siempre se efectúa entre dos números. Los números binarios que deben procesarse se toman de la memoria de datos, el primero se acumula en el registro de trabajo o registro W (de Work = trabajo), el segundo es el presente en el instante en que se invoca la memoria de datos. Como las operaciones pueden ser encadenadas (cuando el resultado sirve como operando de la siguiente operación, tal como el caso de un producto) el registro W tiene un retorno a la ALU.

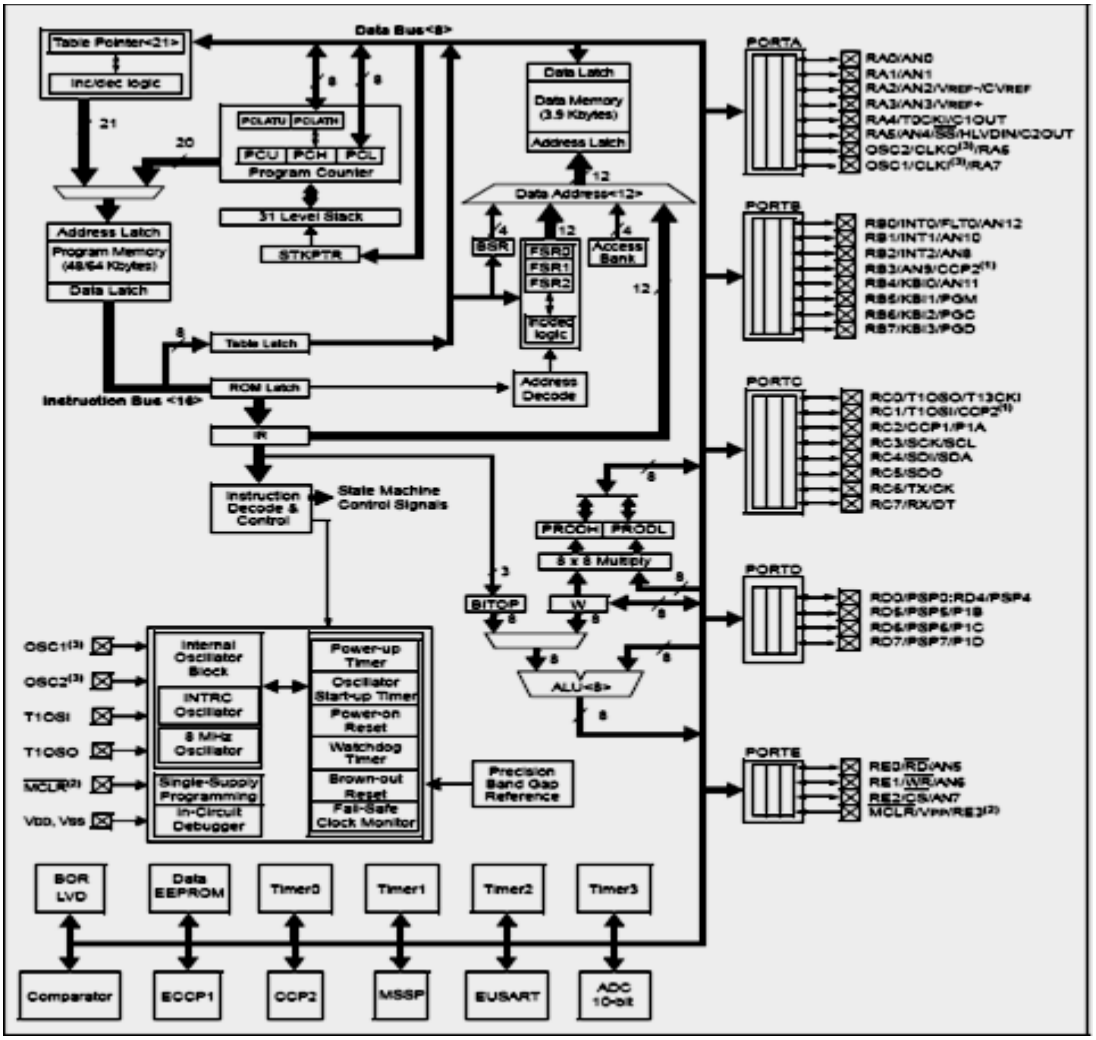


Figura 2. 3 Diagrama de bloques.

Además que la ALU está comandada por el bloque MUX (MUltipleXador). La ALU requiere que se le envíen números para procesar, que llegan desde la memoria de datos, pero antes se la debe predisponer para que efectúe la operación requerida (comparación, rotación de dígitos, etc.)

El registro de estado o estatus colabora durante las operaciones matemáticas. Para realizar una resta: primero ubica el primer número, luego el segundo y después comienza a analizar los bits menos significativos (las unidades), pero si el número de arriba es menor que el número de abajo, entonces lo toma de la columna de las decenas, luego debe recordar esto porque el número de arriba en la columna de las decenas se redujo en una unidad. En realidad, aunque se trate de una operación entre dos números, su ejecución requiere guardar lo que se llama acarreo en otro registro y éste no es otra cosa más que el registro

STATUS, el que informa sobre el estado de la unidad aritmeticológica, el orígen de RESET y los bits de selección del banco de memoria RAM. Cada bit de este registro tiene su función específica. Por ejemplo, para saber si el resultado de una suma ha superado el rango de valores de un byte (es decir, si es mayor de 255) se consulta el estado del bit Carry. Si dicho bit está a 1, se habrá superado el rango; y si está a 0, no se lo supera.

2.3 HERRAMIENTAS DE DESARROLLO.

Se llaman herramientas de desarrollo a aquellas herramientas hardware o software que permiten desarrollar un sistema de control basado en microcontrolador. Estas herramientas son:

- **Simulador**. Es un software que se ejecuta en el PC, y que permite simular la ejecución del programa que se ha elaborado sin necesidad de grabarlo en el microcontrolador. Permite corregir errores antes de programar el microcontrolador.

- **Programador**. Es un equipo que sirve para introducir el programa en la memoria de programa del microcontrolador. Este programador suele funcionar conectado a un PC por el puerto serie, paralelo o usb.
  Desde el software del PC se escribe el programa, se ensambla y se envía al programador, que lo graba en el microcontrolador.

- **Depurador residente**. Consiste en un programa monitor que se encuentra en una parte de la memoria ROM del microcontrolador, y que permite comunicar éste con un PC para poder monitorizar el estado de las diversas variables mientras se ejecuta el programa que se ha grabado en el microcontrolador. Permite depurar los errores que contiene el programa. El inconveniente que tiene es que ocupa una parte de la memoria de programa (normalmente pequeña), y que utiliza una interrupción. Esta interrupción no puede ser utilizada por el programa de usuario.

  Por otra parte, si el programa que se está probando utiliza otras interrupciones, el funcionamiento puede no ser exactamente igual al del sistema final, debido a la presencia del programa monitor (que interrumpe cada cierto tiempo al programa de usuario para comunicar con el PC).

- **Emulador en tiempo real**. Es un circuito electrónico complejo (que tiene un microprocesador más potente que el que se está programando) que emula totalmente el comportamiento del microcontrolador. A este emulador se le pasa el programa, y éste reproduce exactamente lo que haría el microcontrolador en el circuito en el que se conectará (para ello dispone de un terminal con el patillaje idéntico al del microcontrolador).

  Al mismo tiempo permite monitorizar el estado de todas las variables, por lo que permite depurar errores cómodamente. El mayor inconveniente de los emuladores suele ser su elevado precio.

2.4 PROTEUS.[19]

El programa Proteus es una aplicación CAD que se compone de tres módulos básicos:

- **ISIS** (Intelligent Shematic InputSystem) Que es el módulo de captura de esquemas.
- **VSM** (Virtual System Modelling) Es el módulo de simulación, incluyendo PROSPICE.
- **ARES** (Advanced Routing Modelling) Es el módulo para la realización de circuitos impresos.

El módulo ISIS, es un programa que permite dibujar sobre un área de trabajo un circuito que posteriormente se podrá simular. Las utilidades que posee este software son entre otras:

- ✓ Librería de componentes
- ✓ Conexionado automático entre dos puntos
- ✓ Compatibilidad con la mayoría de programas de realización de PCB
- ✓ Enumeración automática de componentes.

Una vez abierto el programa, la pantalla se presenta tal como se indica en la Figura 2.4.

[19] http://www.ieeproteus.com/

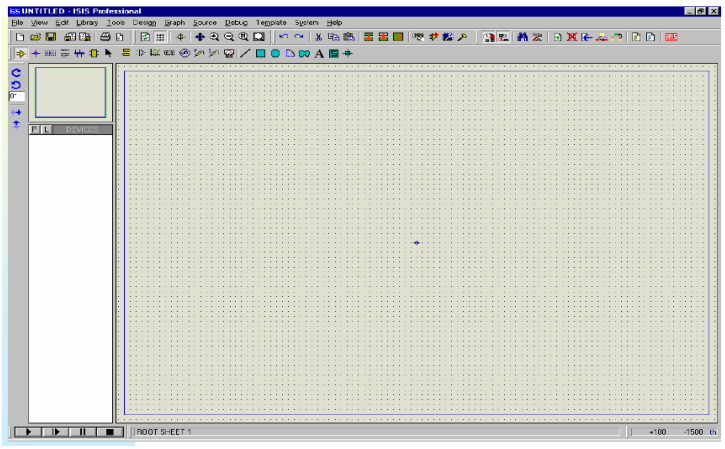


Figura 2. 4 Presentación inicial de la pantalla ISIS

En la Figura 2.5 se detalla cada una de las barras que posee la ventana.

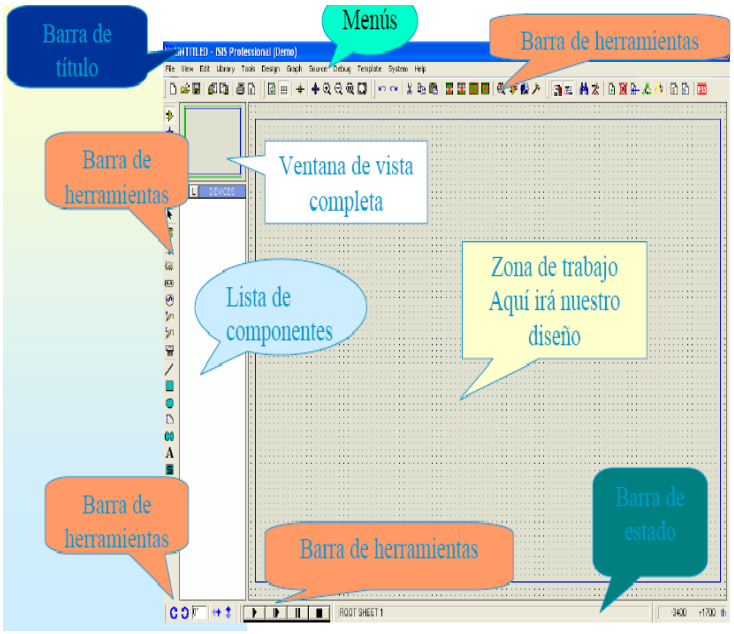


Figura 2. 5 Barras de la ventana principal de ISIS.

2.4.1 El Módulo ISIS.

- **Barra de titulo**.

  Situada en la parte superior de la pantalla, en ella se muestra el icono del programa, el nombre del fichero abierto (Apuntes), la leyenda ISIS PROFESIONAL, y en

ocasiones mensajes indicando que el programa ha entrado en un modo particular de funcionamiento, por ejemplo (Animating) cuando se simula (Figura 2.6).

Figura 2. 6 Barra de título.

La pantalla se divide en:

- **Barra de menús**.

  Permite el acceso a la mayor parte de opciones del programa; sin embargo, algunas opciones están únicamente disponibles en los iconos de la barras de herramientas. (Figura 2.7)

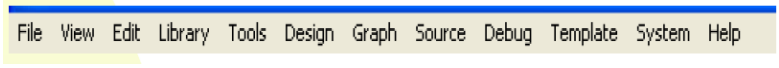


Figura 2. 7 Barra de menús.

- **Barras de herramientas**.

  Son numerosas, y se las puede colocar en cualquiera de sus bordes (Figura 2.8).

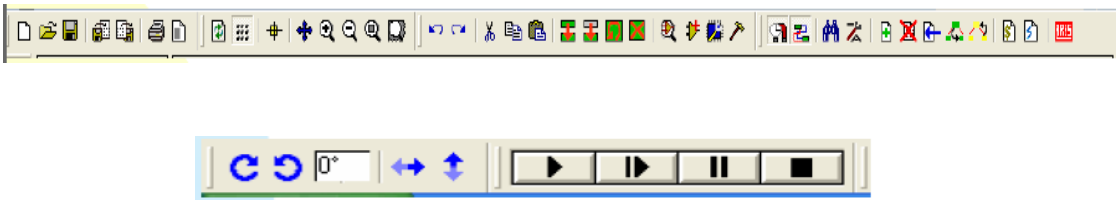


Figura 2. 8 Barra de herramientas.

Una vez realizado el proyecto, presentará un aspecto como el observado en la Figura 2.9.

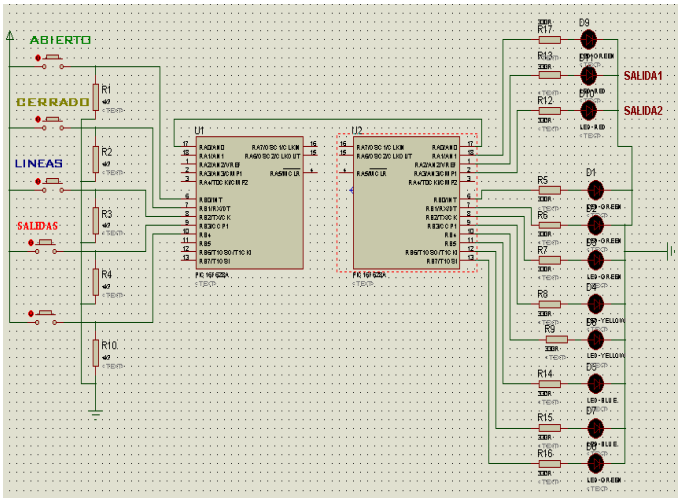


Figura 2. 9 Presentación de un proyecto en ISIS.

- **Ventana de Dispositivos.**

  En esta ventana aparecerán todos los componentes, terminales pines, generadores, etc., que se desee introducir en el esquema, esta ventana dispone de dos botones, que permite acceder a las librerías de componentes incluidas en ISIS (Figura 2.10).

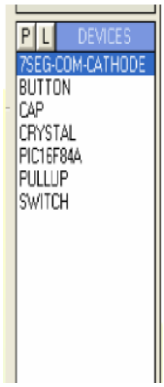


Figura 2. 10 Ventana de dispositivos.

- **Rejilla:** En la construcción de esquemas resulta de gran ayuda disponer de una rejilla en la ventana de edición, además se puede fijar el "paso" de dicha rejilla.
- **Enable/disable grid dot display:** Permite visualizar u ocultar la rejilla que aparece en la ventana de edición.

En la Figura 2.11 se puede observar, cuando la rejilla esta activada y desactivada.

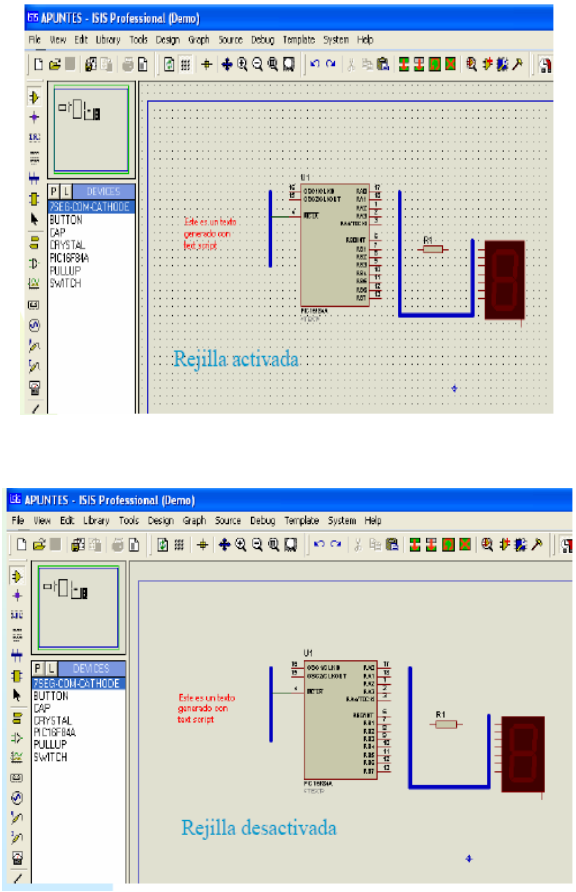


Figura 2. 11 Rejillas en la pantalla principal de ISIS.

2.4.2 Modos de Trabajo en ISIS.

La barra de herramientas de la izquierda permite seleccionar en cada instante el modo de trabajo en el que opera ISIS. Al hacer clic en el modo de trabajo, éste queda seleccionado y su icono muestra una visualización tipo "botón pulsado". El modo de trabajo por defecto, y que más se utiliza es el modo componente, el resto de los modos excepto el de edición instantánea, se utilizan ocasionalmente para poner puntos de unión, etiquetas o buses.

Component / modo componente 

Permite la colocación y orientación de cualquier componente que se encuentre en la ventana de dispositivos; además de permitir la edición de componentes que se encuentren

dentro de la ventana de edición (diagrama esquemático), como se puede observar en la Figura 2.12.

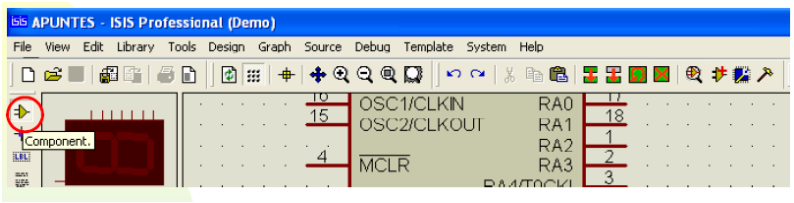


Figura 2. 12 Modo componente.

2.4.3 Captura de Circuitos Electrónicos con el Programa Proteus.

El proceso de captura de esquemas de circuitos electrónicos en ISIS consiste en realizar las siguientes tareas:

- Elegir en las librerías de componentes todos aquellos elementos que se utilizan en el circuito a realizar.
- Situar espacialmente los componentes que forman el circuito en la hoja de trabajo.
- Conectar los terminales de los componentes entre sí.
- Editar las propiedades de los componentes utilizados: valores nominales encapsulados, etc.

2.4.4 Selección de los Componentes del Diseño.

Para seleccionar los componentes que forman el esquema, basta pulsar la tecla P ó en la ventana de dispositivos, pulsar:  Con lo cual se presenta la ventana de *Pick Devices*. Para incluir un componente se realiza los pasos siguientes como se indica en la Figura 2.13.

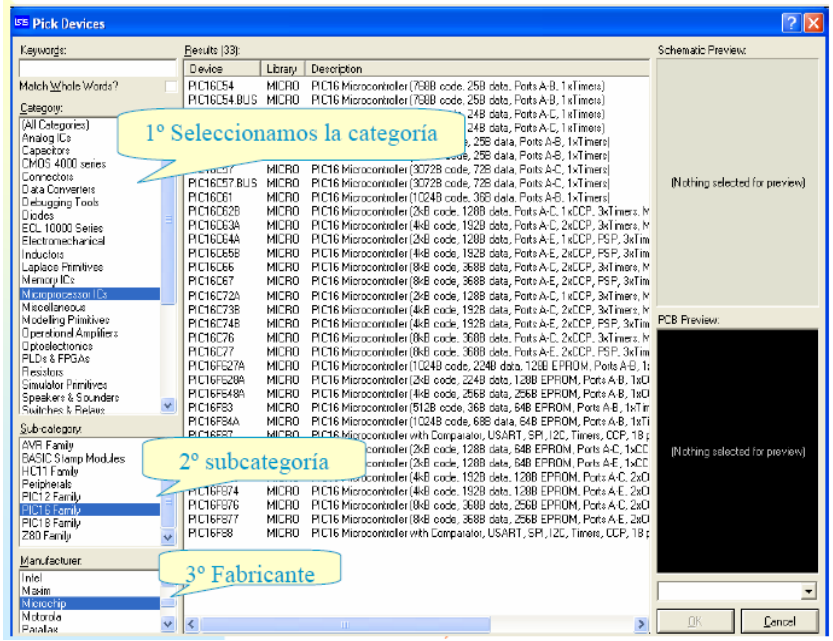


Figura 2. 13 Selección de componentes.

Por último se selecciona el dispositivo, haciendo doble clic (Figura 2.14), como se puede verificar se presenta información sobre el dispositivo como modelo, memoria ROM, RAM, número de puertos, etc.

Seguidamente, se insertan otros componentes; por ejemplo, se desea insertar los siguientes elementos: Condensador (capacidad), Pulsador, Resistor, Switch, cristal.

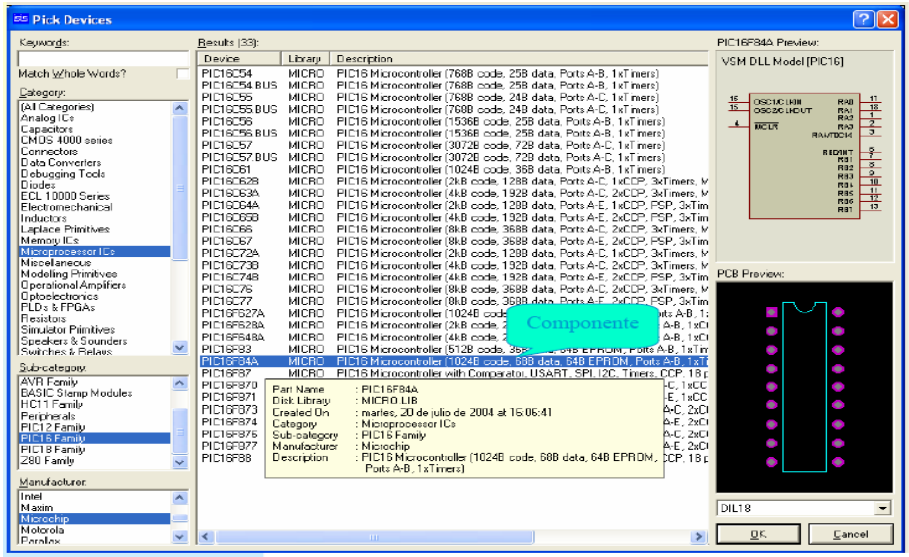


Figura 2. 14 Selección de dispositivo.

Para insertar el capacitor, el proceso para su selección será el siguiente:

1. Se pulsa P en la ventana de dispositivos.

2. Se escribe capacitor en el campo Keywords.

El resultado se puede observar en la Figura 2.15.

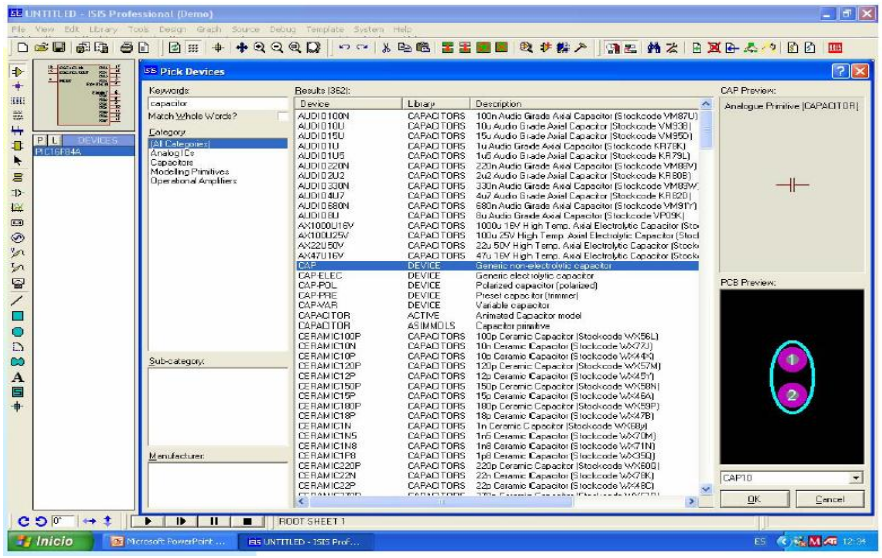


Figura 2. 15 Inserción de componentes.

- Los modelos de simulación *AnaloguePrimitive*se utilizan en las simulaciones de PROSPICE y en las animaciones VSM, mientras que los modelos de simulación *SchematicModel*se utilizan en las animaciones con VSM.

- Como se ha indicado, haciendo doble clic, el componente se añade a la lista de dispositivos.

- Así pues se añade todos los componentes antes enumerados a la ventana de dispositivos.

Una vez incluidos todos los componentes en la ventana de dispositivos, se tendrá una pantalla como se indica en la Figura 2.16.

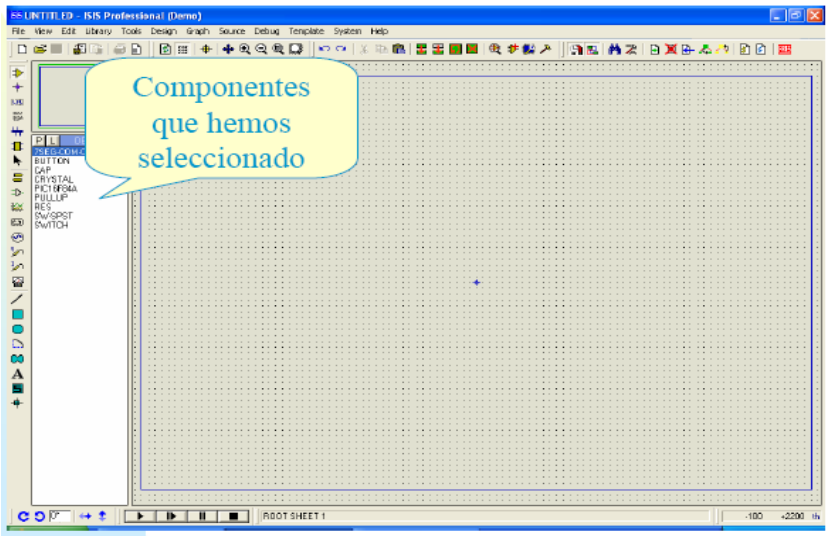


Figura 2. 16 Componentes seleccionados.

2.4.5 Inserción de Componentes en la Hoja de Trabajo.

Se debe tener activo el modo componente: . Además deben estar activas *Real Time Anotation* (numeración de los componentes en tiempo real) y *WireAuto Router* (trazado de hilos automáticos) dentro del menú *TOOLS* (herramientas), como se observa en la Figura 2.17.

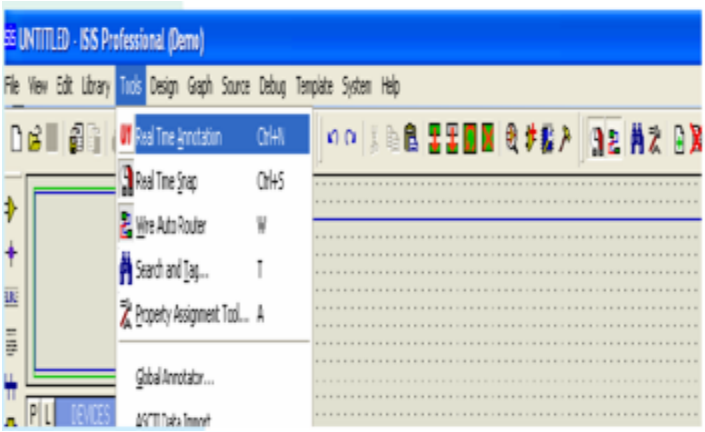


Figura 2. 17 Inserción de componentes.

Para situar un componente en la hoja de trabajo, se debe seleccionarlo en la ventana de dispositivos (Devices) y hacer clic en la ventana de edición (Figura 2.18).

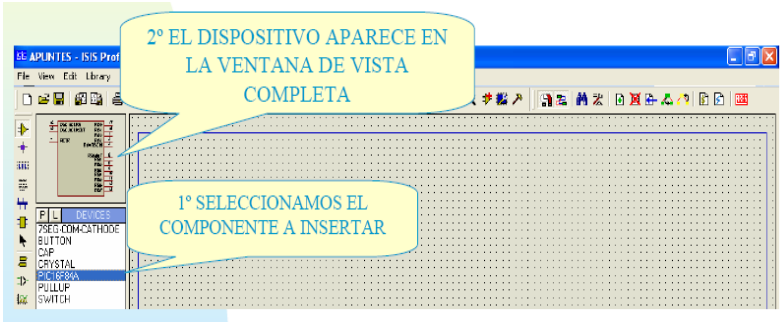


Figura 2. 18 Selección de componentes.

Una vez que el componente se encuentra seleccionado, cada vez que se haga clic en la ventana de trabajo, se insertará este componente, como se indica en la Figura 2.19.

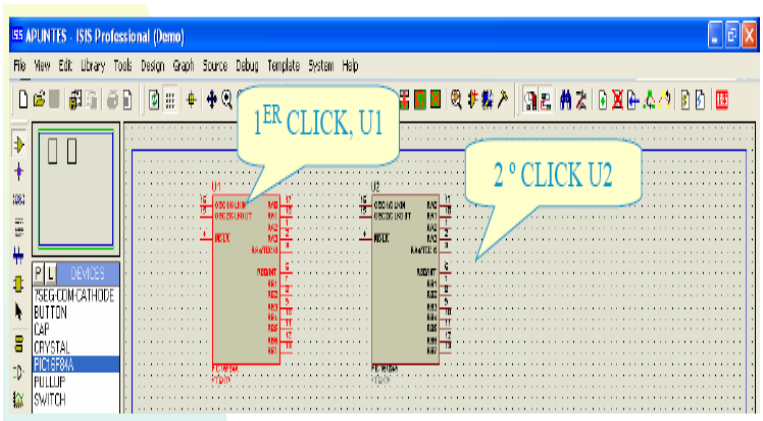


Figura 2. 19 Inserción de un microcontrolador.

2.4.6 Edición de las Propiedades de los Componentes.

Existen dos maneras de editar un componente. Si se ha seleccionado el modo de componente en la barra de herramientas, los pasos son:

- Se selecciona el elemento a editar (botón derecho del ratón).
- Se da clic con el ratón sobre el elemento ya seleccionado.

Si se ha seleccionado el modo de edición instantánea en la barra de herramientas, bastará hacer clic sobre el elemento a editar, este modo es cómodo cuando ya se ha insertado todos los componentes sobre la hoja de trabajo.

Independientemente del proceso que se siga, si se edita por ejemplo el PIC, se obtendrá una ventana como la que sigue en la Figura 2.20.

En la ventana se pueden definir entre otras las siguientes propiedades:

- La referencia del componente (*ComponentReference).*
- El valor del componente (*Componentvalue).*
- El encapsulado (*PCB Package).*

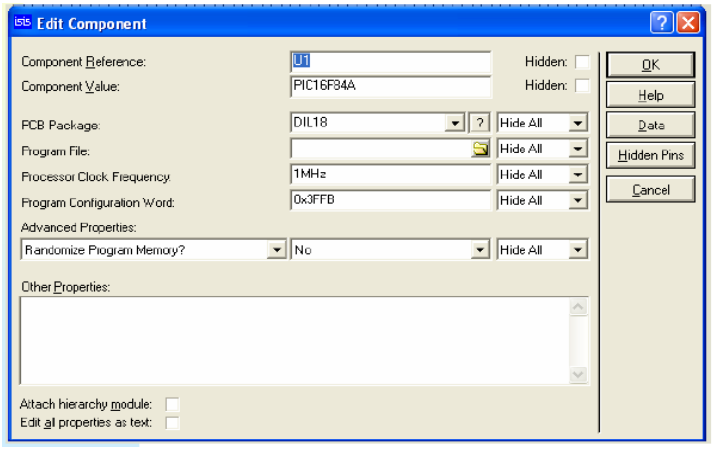


Figura 2. 20 Edición de un PIC.

Además se puede indicar:

- El fichero con el contenido del programa fuente *(ProgramFile)*
- La frecuencia de trabajo (*ProcesorClockFrecuency)*
- La palabra de configuración (*ProgramConfigurationWord)*
- Etc.

2.4.7 Conexionado de los Componentes del Diseño.

Una vez colocados todos los componentes en la zona de edición se pasa a unirlos eléctricamente. Las conexiones entre los terminales de los componentes dispuestos en la hoja de trabajo se pueden realizar mediante hilos (*wire*) y uniones (*junctiondot*) o bien mediante etiquetas (*labe*l).

Si se coloca el puntero encima de un terminal de cualquier componente, se comprueba que dicho puntero se transforma en una X indicando que se puede unir a otro terminal, tal como se indica en la Figura 2.21.

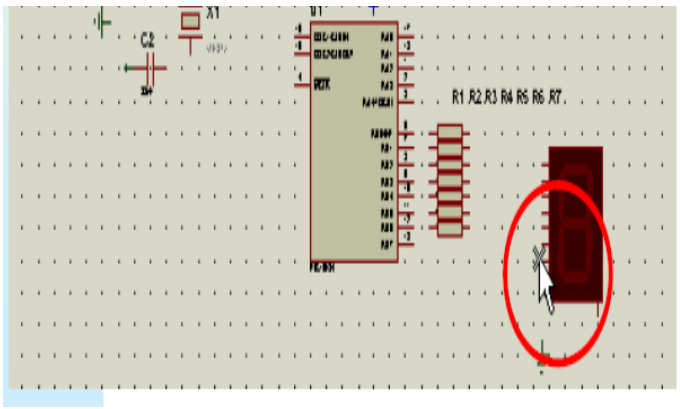


Figura 2. 21 Conexión de componentes.

En la Figura anterior, si se pulsa el botón izquierdo del ratón y se lo mueve se irá generando un conexión eléctrica, que aparece de color rosa, como indica la Figura 2.22.

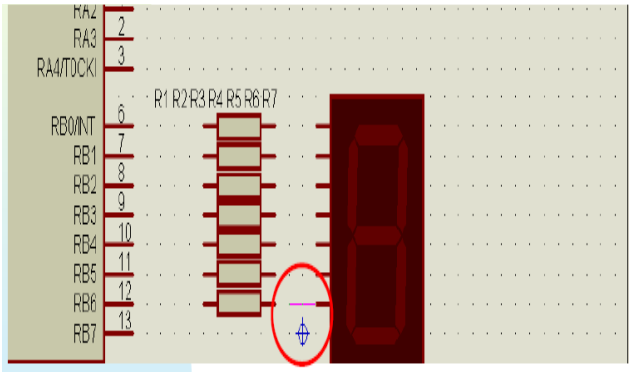


Figura 2. 22 Conexión eléctrica.

Cuando se alcance un terminal de otro componente; se indicará, ya que el cursor vuelve a cambiar en X, si en esta situación se da clic en el botón izquierdo del ratón, la conexión eléctrica quedará realizada, pasando a color verde (Figura 2.23).

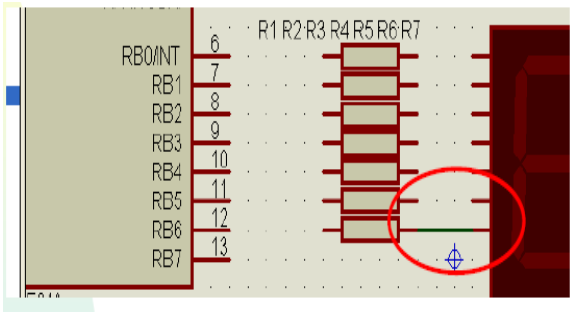


Figura 2. 23 Conexión realizada.

Si se elige como principio o final de un hilo otro hilo, ISIS dibuja automáticamente el punto de unión necesario entre ambos hilos, como se indica en la Figura 2.24.

Una vez dibujados los hilos, éstos se pueden seleccionar y una vez seleccionados se pueden borrar, mover, etc.

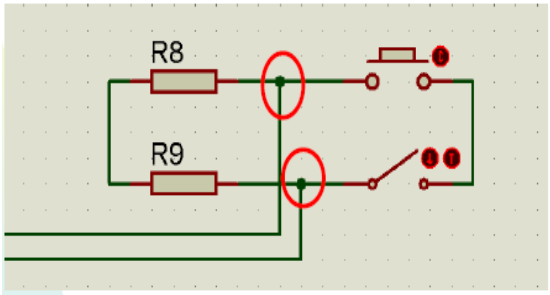


Figura 2. 24 Punto de unión entre hilos.

2.5 ARES.[20]

2.5.1 Presentación de ARES.

Es una herramienta de la suite Proteus dedicada al diseño de placas de circuito impreso (PCB). Está plenamente integrada con la herramienta **ISIS.**

Una vez diseñado en **ISIS** el esquema electrónico, se genera automáticamente la lista de redes (NETLIST). Una red es un grupo de pines interconectados entre sí y la lista de redes es una lista con todas las redes que forman el diseño. **ARES** es capaz de recibir esta lista de redes para diseñar, a partir de ella, la placa de circuito impreso. De esta forma se

[20] http://www.ieeproteus.com/ares.html

asegura que la placa tendrá unidos entre sí los pines de forma idéntica a como se ha definido en el esquema electrónico.

Cualquier modificación que se realice en el esquema, podrá ser reenviado desde **ISIS** a **ARES** donde aparecerán resaltados los cambios que se hayan producido. De esta forma la modificación y rediseño de la placa se realizará de forma mucho más simple y segura.

Al igual que **ISIS** era capaz de enviar su listas de redes a programas de terceras partes, también **ARES** es capaz de importar listas de redes procedentes de otros programas que utilicen los formatos estándar Tango o Multiwire.

2.5.2 Uso de Base de Datos de Redes de Alta Resolución 32bits.

ARES es una herramienta que facilita la realización de los más complejos diseños de circuitos impresos, gracias a la utilización de una base de datos de redes de alta resolución de 32 bits. Con esta técnica se posibilita la colocación de elementos con una resolución lineal en placas de circuito impreso de hasta 20 cm. Además, cualquier componente puede ser rotado con una resolución angular de 0,1 grados. Por todo ello, Ares es una herramienta ideal para realizar placas con trazados de pistas, como se puede ver en la Figura 2.25.

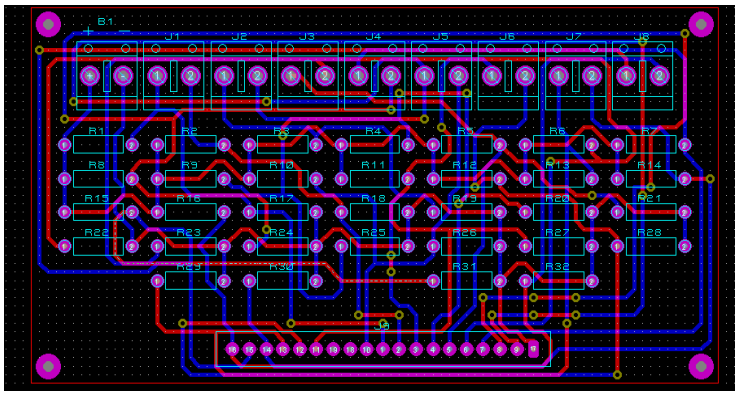


Figura 2. 25 Pantalla ARES.

2.5.3 Generación Automática de Pistas.

ARES incorpora un avanzado trazador automático de pistas basado en rejilla. Su potencia, rapidez y flexibilidad permite generar todas las rutas de la placa de circuito impreso con

pistas de cualquier grosor, utilizando vías de cualquier ancho, a 90 o 45 grados y gestionando desde una a ocho capas.

La selección del rango del tamaño de la rejilla utilizada permite al usuario alcanzar un compromiso entre la densidad de pistas y la velocidad de ejecución de la herramienta de trazado automático. Es posible alcanzar incluso densidades de dos y tres pistas entre las diferentes coronas (pads) de un circuito integrado (Figura 2.26).

El trazador automático de pistas incorpora rutinas especiales para el tratamiento de elementos SMD con separación entre pines menores a 25 o 50 milésimas de pulgadas. Si no fuera por estas rutinas especiales, los pines quedarían situados fuera de la rejilla base y no se podrían unir las pistas a las coronas de cada patilla del componente.

La técnica conocida con el nombre de "romper y reintentar" (rip-up & retry) permite borrar y recolocar las pistas de forma automática, permitiendo completar la operación de auto-trazado en la mayoría de las tarjetas de densidad media con pistas de 25 a 50 milésimas de pulgada.
Finalmente, se puede ordenar ejecutar una última operación de mejora denominada tidy-pass que reduce ambos, la longitud de las pistas y el número de vías utilizadas. Con ella, además, se logran mejoras sustanciales en la calidad estética de la placa de circuito impreso.

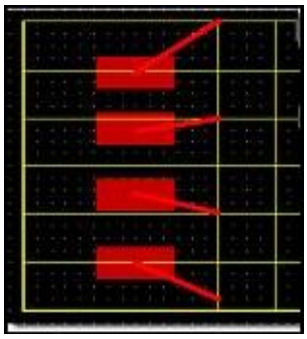


Figura 2. 26 Ejemplo de Trozado de pistas.

2.5.4 Trazado y Modificado Manual de Pistas.

Para trazar una nueva pista no es necesario seguir los trazos de unión existentes. ARES permite colocar pistas donde se desee. Cuando ARES detecte que dos pines han sido correctamente enlazados con una pista, automáticamente borrará el trazo de unión correspondiente.

Cuando se desee modificar una pista existente, se puede redibujar una nueva ruta para esa pista o borrar cualquier trozo de pista existente, como se observa en la Figura 2.27, independientemente del método utilizado para su creación. Comandos especiales están disponibles para modificar de forma sencilla el ancho de cualquier pista o para moverla de una capa a otra.

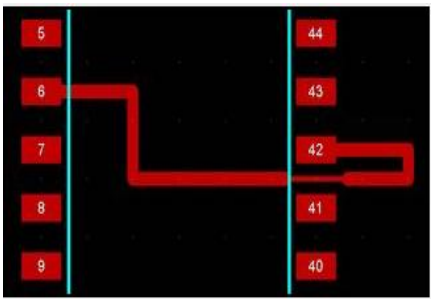


Figura 2. 27 Modificación de una pista.

Si al trazar manualmente una pista entre dos obstáculos, como por ejemplo, dos coronas de un circuito integrado, ARES detecta el ancho de esa pista y provoca que se supere la distancia mínima de separación fijada entre pistas y coronas. ARES, automáticamente, crea un cuello más estrecho en la pista para cumplir las reglas de diseño.

El trazado de pistas con forma de curva es tan sencillo, como presionar la tecla CTRL, mientras se genera la pista con el ratón. El radio y la distancia del arco se modificarán automáticamente para adaptarse a la posición donde se situe el puntero del ratón, como se indica en la Figura 2.28.

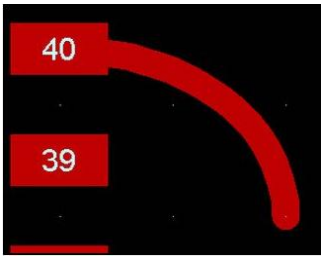


Figura 2. 28 Curvas en una pista.

2.5.5 Presentación de los Circuitos en 3D.

Ares incluye un motor de presentaciones en 3D (Figura 2.29), en las versiones más altas (nivel 2 o superior), posibilitando la visualización en tres dimensiones de la tarjeta antes de realizar el prototipo. Esta capacidad no solo ayuda en el diseño de la placa, sino que también proporciona información de su posible altura.

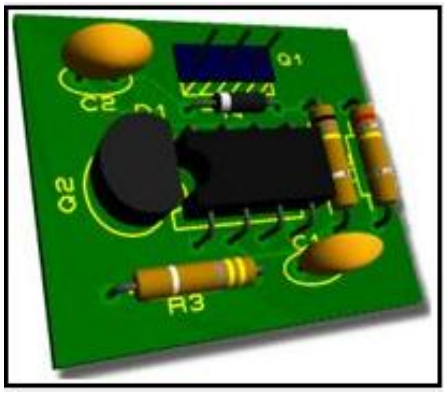


Figura 2. 29 Presentación en 3D.

Se puede disponer de diferentes vistas de la imagen visualizada, girarla, hacer zoom sobre ella, etc. También se puede seleccionar el nivel de detalle de la visualización.

Además de poder imprimir el resultado final del trabajo utilizando las impresoras definidas en Windows; ARES dispone de un driver optimizado HPGL para ploters de plumilla y un conjunto completo de herramientas para los fabricantes de placas de circuito impreso.

2.6 CONSIDERACIONES BÁSICAS PARA MONTAR UN PROYECTO.[21]

En todo proceso industrial se requiere tener bajo control una o varias magnitudes de salida, a partir de la información de entrada que esté presente en el sistema, tomando en cuenta en todo momento una lógica de seguridad intrínseca propia del sistema.

Los PLC pueden controlar procesos de cualquier tipo, ya sean industriales o de uso no tan riguroso, dependiendo de las siguientes necesidades:

- Espacio reducido.
- Procesos de producción periódicamente cambiantes.
- Procesos secuenciales.
- Maquinaria de procesos variables.
- Verificación de las distintas partes del proceso de forma centralizada.

2.6.1 Nociones Básicas para el Montaje del PLC.

Para realizar un proyecto se debe conocer las características técnicas de los elementos que se utilizará para realizar el mismo.
A continuación se detalla funciones básicas y características de los principales elementos utilizados en el PLC.

- **RELES.**[22]

Un relé es un sistema mediante el cual se puede controlar una potencia mucho mayor con un consumo en potencia muy reducido. Su funcionamiento se basa en el fenómeno electromagnético. Cuando la corriente atraviesa la bobina, produce un campo magnético que magnetiza un núcleo de hierro dulce (ferrita). Éste atrae al inducido que fuerza a los contactos a tocarse. Cuando la corriente se desconecta vuelven a separarse.

Las aplicaciones de este tipo de componentes son múltiples: en electricidad, en automatismos eléctricos, control de motores industriales; en electrónica: sirven

[21] Reyes Carlos. Microcontroladores PIC. Edición II. 2007, Pag 17.
[22] http://www.wikiciencia.org/electronica/electricidad/reles/index.php

básicamente para manejar tensiones y corrientes superiores a los del circuito propiamente dicho, se utilizan como interfaces para PC como se observa en la Figura 2.30; en interruptores crepusculares, en alarmas, en amplificadores, etc.

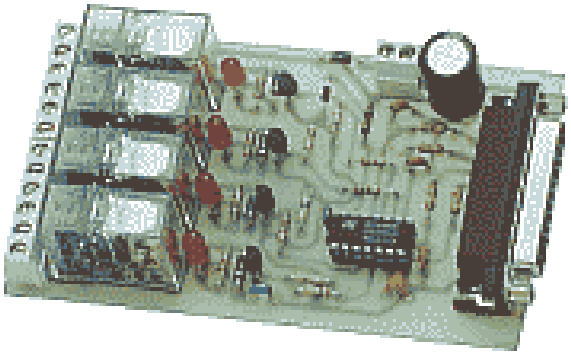


Figura 2. 30 Aplicación de los relés como módulos de interface.

Tipos de relés.

- Relés electromecánicos:

  Convencionales.
  Polarizados.
  Reed inversores.

- Relés híbridos.
- Relés de estado sólido.

Estructura de un relé.

En general, se puede distinguir en el esquema de un relé los siguientes bloques:

- Circuito de entrada, control o excitación.
- Circuito de acoplamiento.
- Circuito de salida, carga o maniobra, constituido por:

  - circuito excitador.
  - dispositivo conmutador de frecuencia.
  - protecciones.

Se puede observar un relé y su estructura en la Figura 2.31.

Figura 2. 31 Estructura de un relé.

Características generales.

Las características generales de cualquier relé son:

- El aislamiento entre los terminales de entrada y de salida.
- Adaptación sencilla a la fuente de control.
- Posibilidad de soportar sobrecargas, tanto en el circuito de entrada como en el de salida.
- Las dos posiciones de trabajo en los bornes de salida de un relé se caracterizan por:

  - En estado abierto, alta impedancia.
  - En estado cerrado, baja impedancia.

A los relés de estado sólido se pueden añadir otras características:

- Gran número de conmutaciones y larga vida útil.
- Conexión en el paso de tensión por cero, desconexión en el paso de intensidad por cero.
- Ausencia de ruido mecánico de conmutación.
- Escasa potencia de mando, compatible con TTL y MOS.
- insensibilidad a las sacudidas y a los golpes.

- Cerrado a las influencias exteriores por un recubrimiento plástico.

Relés de tipo armadura.

Son los más antiguos y también los más utilizados. La Figura 2.32 explica prácticamente su constitución y funcionamiento. El electroimán hace vascular la armadura al ser excitada, cerrando los contactos dependiendo de si es N.O ó N.C (normalmente abierto o normalmente cerrado).

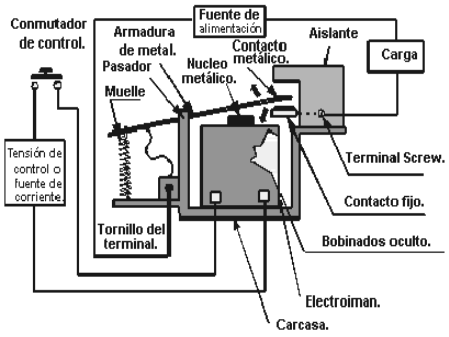


Figura 2. 32 Relé tipo armadura.

- **DIODOS.**[23]

Un diodo es un dispositivo semiconductor que permite el paso de la corriente eléctrica en una única dirección con características similares a un interruptor; se les suele denominar rectificadores, ya que son dispositivos capaces de convertir una corriente alterna en corriente continua.

A la hora de elegir un diodo para una aplicación concreta se debe cuidar que presente unas características apropiadas para dicha aplicación. Para ello, se debe examinar cuidadosamente la hoja de especificaciones (Tabla 2.2 – 2.3), que el fabricante provee.

[23] http://es.wikipedia.org/wiki/Diodo

PRIMARY CHARACTERISTICS	
$I_{F(AV)}$	1.0 A
V_{RRM}	50 V to 1000 V
I_{FSM} (8.3 ms sine-wave)	30 A
I_{FSM} (square wave t_p = 1 ms)	45 A
V_F	1.1 V
I_R	5.0 µA
T_J max.	150 °C

Tabla 2. 2 Hoja de características del diodo 1N4007.

MAXIMUM RATINGS (T_A = 25 °C unless otherwise noted)									
PARAMETER	SYMBOL	1N4001	1N4002	1N4003	1N4004	1N4005	1N4006	1N4007	UNIT
Maximum repetitive peak reverse voltage	V_{RRM}	50	100	200	400	600	800	1000	V
Maximum RMS voltage	V_{RMS}	35	70	140	280	420	560	700	V
Maximum DC blocking voltage	V_{DC}	50	100	200	400	600	800	1000	V
Maximum average forward rectified current 0.375" (9.5 mm) lead length at T_A = 75 °C	$I_{F(AV)}$	1.0							A
Peak forward surge current 8.3 ms single half sine-wave superimposed on rated load	I_{FSM}	30							A
Non-repetitive peak forward surge current square waveform T_A = 25 °C (Fig. 3) t_p = 1 ms / t_p = 2 ms / t_p = 5 ms	I_{FSM}	45 35 30							A
Maximum full load reverse current, full cycle average 0.375" (9.5 mm) lead length T_L = 75 °C	$I_{R(AV)}$	30							µA
Rating for fusing (t < 8.3 ms) (1)	I^2t	3.7							A^2s
Operating junction and storage temperature	T_J, T_{STG}	- 50 to + 150							°C

Tabla 2. 3 Hoja de características de diodos.

Las características comerciales más importantes de los diodos que aparecen en cualquier hoja de especificaciones son:

1. Corriente máxima en directa, IFmax o IFM (DC forward current): Es la corriente continua máxima que puede atravesar el diodo en polarización directa sin que éste sufra ningún daño, puesto que una alta corriente puede provocar un calentamiento por efecto Joule excesivo. Los fabricantes suelen distinguir tres límites:

    - Corriente máxima continua (IFM)
    - Corriente de pico transitoria (Peak forward surge current), en la que se especifica también el tiempo que dura el pico
    - Corriente de pico repetitivo (Recurrent peak forward current), en la que se especifica la frecuencia máxima del pico

2. Tensión de ruptura en polarización inversa (Breakdown Voltage, BV; Peak Inverse Voltage, PIV): Es la tensión a la que se produce el fenómeno de ruptura por avalancha.
3. Tensión máxima de trabajo en inversa (Maximun Working Inverse Voltage): Es la tensión que el fabricante recomienda no sobrepasar para una operación en inversa segura.
4. Corriente en inversa, IR (Reverse Current): Es habitual que se exprese para diferentes valores de la tensión inversa.
5. Caída de tensión en PD, VF (Forward Voltage): Pese a que se ha señalado anteriormente los 0.7V como valor típico, en muchas ocasiones los fabricantes aportan datos detallados de esta caída de tensión, mediante la gráfica I-V del dispositivo.

Se define como el voltaje mínimo que necesita un diodo para entrar en conducción.
Para los diodos de silicio el voltaje de umbral debe ser de 0,6V aproximadamente, mientras que para los de germanio debe ser de 0,2V aproximadamente.
De acuerdo a esta regla, con voltajes inferiores a los indicados no se establece corriente a través de los mismos.

Uno de los diodos más utilizados es el rectificador de silicio. Estos dispositivos admiten un amplio rango de corrientes, desde décimas hasta varios centenares de amperes, pudiendo operar con tensiones de 1000 V o más y pueden funcionar algunos de estos dispositivos con temperaturas ambientes de hasta 200 °C.

Cuando son utilizados correctamente alcanzan una larga vida útil, ya que no los afectan la acción del tiempo, la humedad y la temperatura.

- **Corriente máxima (I_{max}).**

  Es la intensidad de corriente máxima que puede conducir el diodo sin fundirse por el efecto Joule. Dado que es función de la cantidad de calor que puede disipar el diodo, depende sobre todo del diseño del mismo.

- **Corriente inversa de saturación (I_s).**

Es la pequeña corriente que se establece al polarizar inversamente el diodo por la formación de pares electrón-hueco debido a la temperatura, admitiéndose que se duplica por cada incremento de 10º en la temperatura.

- **Corriente superficial de fugas.**

Es la pequeña corriente que circula por la superficie del diodo, esta corriente es función de la tensión aplicada al diodo, con lo que al aumentar la tensión, aumenta la corriente superficial de fugas.

- **Tensión de ruptura (PIV).**

Es la tensión inversa máxima que el diodo puede soportar antes de darse el efecto avalancha.

Teóricamente, al polarizar inversamente el diodo, éste conducirá la corriente inversa de saturación; en la realidad, a partir de un determinado valor de la tensión, en el diodo *normal* o de *unión abrupta* la ruptura se debe al efecto avalancha.

- **TTL.**[24]

TTL es la sigla en inglés de *Transistor-Transistor Logic* o "Lógica Transistor a Transistor". Es una familia lógica o lo que es lo mismo, una tecnología de construcción de circuitos electrónicos digitales. En los componentes fabricados con tecnología TTL los elementos de entrada y salida del dispositivo son transistores bipolares (Figura 2.33).

Figura 2. 33 TTL 74LS164N.

[24] es.wikipedia.org/wiki/Tecnología_TTL

Características.

- Su tensión de alimentación característica se halla comprendida entre los 4,75v y los 5,25V.
- Los niveles lógicos vienen definidos por el rango de tensión comprendida entre 0,2V y 0,8V para el estado L (bajo) y los 2,4V y Vcc para el estado H (alto).
- La velocidad de transmisión entre los estados lógicos es muy buena, si bien esta característica hace que aumente su consumo. Motivo por el cual han aparecido diferentes versiones de TTL como FAST, LS, S, etc.
- Las señales de salida TTL se degradan rápidamente si no se transmiten a través de circuitos adicionales de transmisión (no pueden viajar más de 2 m por cable sin graves pérdidas).

Los circuitos de tecnología TTL se prefijan normalmente con el número 74 (54 en las series militares e industriales). A continuación un código de una o varias cifras que representa la familia.

Con respecto a las familias cabe distinguir:

- TTL : Serie estándar
- TTL-L (low power) : Serie de bajo consumo
- TTL-S (schottky) : Serie rápida (usa diodos Schottky)
- TTL-AS (advanced schottky) : Versión mejorada de la serie anterior
- TTL-LS (low power schottky) : Combinación de las tecnologías L y S (es la familia más extendida)
- TTL-ALS (advanced low power schottky) : Versión mejorada de la serie AS
- TTL-F (FAST : fairchild advanced schottky)
- TTL-AF (advanced FAST) : Versión mejorada de la serie F
- TTL-HC (high speed C-MOS) : Realmente no se trata de tecnología TTL bipolar sino CMOS
- TTL-HCT (high speed C-MOS) : Serie HC dotada de niveles lógicos compatibles con TTL
- TTL-G (GHz C-MOS) : GHz (From PotatoSemi)
- Etapa de entrada por emisor. Se utiliza un transistor multiemisor en lugar de la matriz de diodos de DTL.

- Separador de fase. Es un transistor conectado en emisor común que produce en su colector y emisor señales en contrafase.
- Driver. Está formada por varios transistores, separados en dos grupos. El primero va conectado al emisor del separador de fase y drenan la corriente para producir el nivel bajo a la salida. El segundo grupo va conectado al colector del divisor de fase y produce el nivel alto.

Configuraciones para los TTL.

Esta configuración general varía ligeramente entre dispositivos de cada familia, principalmente la etapa de salida, que depende de si son buffers o no, y si son de colector abierto, tres estados (ThreeState), etc. Mayores variaciones se encuentran entre las distintas familias: 74N, 74L y 74H; difieren principalmente en el valor de las resistencias de polarización, pero la mayoría de los 74LS (y no 74S) carecen del transistor multiemisor característico de TTL. En su lugar llevan una matriz de diodos Schottky (como DTL). Esto les permite aceptar un margen más amplio de tensiones de entrada, hasta 15V en algunos dispositivos, para facilitar su interface con CMOS. También es bastante común, en circuitos conectados a buses, colocar un transistor PNP a la entrada de cada línea, para disminuir la corriente de entrada y así la cargar menos el bus. Existen dispositivos de interface que integran impedancias de adaptación al bus para disminuir la reflexión u aumentar la velocidad.

- **Transistores.**

El transistor es un componente electrónico activo, un dispositivo que puede amplificar una corriente, produciendo una señal de salida mucho más potente que la señal de entrada. Esta potencia adicional proviene de una fuente externa. Es esencial de todo circuito electrónico, desde el amplificador más sencillo hasta la computadora digital más elaborada. Por ejemplo, los circuitos integrados son circuitos construidos con transistores y otros componentes sobre una pequeña pastilla de material semiconductor.
Un transistor es un dispositivo de 3 terminales, disponible en dos variedades: NPN y PNP. En la Figura 2.34, se observan los símbolos que representan a ambos transistores, donde B es la base, C el colector y E el emisor, y se indica el sentido de los flujos de corriente.

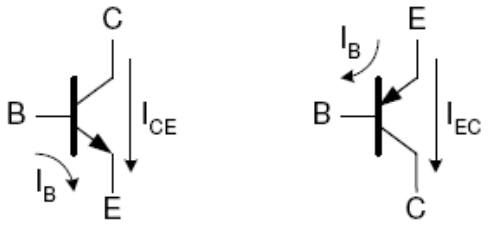


Figura 2. 34 Representación esquemática de un transistor.

En la utilización de un transistor NPN se sigue las siguientes reglas:

- El emisor debe estar a una tensión mayor que el colector.

- Los circuitos base-emisor y base-colector se comportan como diodos. Normalmente, el diodo base-emisor está polarizado en directa, y el base-colector en inversa, es decir, contrario a la dirección de flujo de corriente. La flecha en el símbolo del transistor indica el diodo polarizado en directa.

- Cada transistor tiene un valor máximo de I_C, I_B y V_{CE} que no puede excederse sin correr el riesgo de destruirlo. Hay también otros límites, como la potencia disipada I_C, V_{CE}, temperatura, V_{BE}, señalados en las hojas de datos.

- Cuando se sigue estos pasos, la corriente I_C es proporcional a I_B y puede escribirse como:

$$I_C = \beta I_B \qquad \text{Ec 2.1}$$

Donde β es la ganancia de corriente. Además:

$$I_E = I_C + I_B = (\beta + 1) I_B \qquad \text{Ec 2.2}$$

Además, β es muy variable de un transistor a otro, incluso del mismo modelo, por ejemplo en el TBJ NPN 2N3904 (Figura 2.35), su valor varía entre 100 y 300, por lo que se vuelve necesario diseñar circuitos cuyo comportamiento sea poco dependiente de su valor.

Tanto I_C, como I_B fluyen hacia el emisor, y esta ganancia proviene de una fuente conectada entre colector y emisor. Esta propiedad da al transistor su utilidad: una corriente pequeña en la base controla una corriente mucho más grande en el colector. La potencia que consume el transistor está dada por:

$$P_D = V_{CE} \cdot I_C \qquad \text{Ec 2.3}$$

Si se conecta los terminales base - emisor a un voltaje mayor que 0,6 a 0,8 V, una corriente enorme circulará por el transistor. Esta regla implica que un transistor operando correctamente tendrá una tensión $V_B \approx V_E + 0.7$ V, donde el último término es la caída de tensión sobre el diodo BE en directa.

En los casos límites de operación (corte y saturación), el transistor actúa como una llave de corriente. Si no hay corriente en la base, el transistor no deja pasar corriente entre colector y emisor, y la tensión entre esos dos terminales es máxima. Si la corriente de base es máxima, también lo será la corriente de colector y emisor, y la tensión entre ellos será mínima.

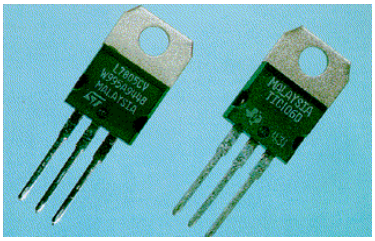


Figura 2. 35 Transistor 2N3904.

Además, debe considerarse que el costo de los TBJ (para aplicaciones afines) suele estar asociado a su capacidad tanto de manejar corriente, así como también de disipar potencia.
De todo esto, se encuentra que el modelo 2N3904 resulta adecuado para este proyecto, ya que cumple con los requisitos anteriores y a su vez no resulta sobredimensionado, a continuación se presenta en la Tabla 2.4 las principales características del transistor.

TBJ NPN 2N3904		
P_D máxima	625	mW
V_{BE} on	0,7	V
V_{CE} máxima	40	
V_{CE} saturac.	0,3	
I_C máxima	200	mA
I_B mínima	0,05	
β mínimo	100	-

Tabla 2. 4 Características del Transistor 2N3904.

- **LM 7805.**

Existen muchas maneras de lograr un voltaje estable, pero en general utilizan varios componentes discretos, lo que redunda en un costo elevado, un diseño más complicado, y circuitos más grandes. La alternativa es utilizar algún regulador de tensión integrado, disponibles para casi todos los voltajes, y para corrientes desde unas pocas centésimas de Amper hasta varios Amperes.

Dentro de los reguladores de voltaje con salida fija, se encuentran los pertenecientes a la familia LM78xx, donde "xx" es el voltaje de la salida. Éstos son 5, 6, 8, 9, 10, 12, 15, 18 y 24V, entregando una corriente máxima de 1 Amper y soporta consumos pico de hasta 2.2 Amperes. Poseen protección contra sobrecargas térmicas y contra cortocircuitos, que desconectan el regulador en caso que su temperatura de juntura supere los 125°C. Los LM78xx son reguladores de salida positiva. Así, un LM7805 es capaz de entregar 5 voltios positivos.

La potencia además depende de la tensión de entrada; por ejemplo, si se tiene un LM7812, cuya tensión de salida (Vout) es de 12 V, con una tensión de entrada (Vint) de 20 V, y una carga en su salida (Iout) de 0,5A, multiplicando la diferencia entre la tensión de entrada y la tensión de salida por la corriente que circulará por la carga, da los vatios que va a tener que soportar el integrado.

$$P = (Vint - Vout) \times Iout \qquad \text{Ec 2.4}$$

$$P = (20 - 12) \times 0.5 = 4W$$

La tensión de entrada es un factor muy importante, ya que debe ser superior en unos 3 voltios a la tensión de salida (el mínimo recomendado por el fabricante), pero todo el exceso debe ser eliminado en forma de calor. Si en el ejemplo anterior en lugar de entrar con 20 volts solo es 15V (los 12V de la salida más el margen de 3V sugerido) la potencia disipada es mucho menor:

$$P= (Vint - Vout) \times Iout = (15 - 12) \times 0.5 = 1.5W$$

En la Figura 2.36 se ve la disposición de pines de estos reguladores. En este caso, el pin 1 corresponde a la entrada (input), el pin 2 es el punto común (common) y el pin3 es el correspondiente a la salida (output). Es importante conectarlos de manera adecuada, ya que es posible que se destruyan si son conectados en forma incorrecta.

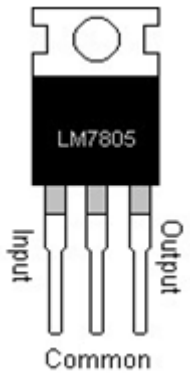


Figura 2. 36 Regulador LM7805.

El voltaje máximo que soportan en la entrada es de 35 voltios para los modelos del LM7805 al 7815.

A continuación se procede a dimensionar algunos elementos asociados al PIC, tal que permitan su correcto funcionamiento.

1. El PIC tiene tecnología CMOS, esto quiere decir que consume muy poca corriente, pero a la vez es susceptible a daños por estática, se recomienda utilizar pinzas para manipular y así poder transportar desde el grabador al protoboard o viceversa, o a su vez utilizar una manilla antiestática.

2. Utilizar un regulador de voltaje como el 7805 (Figura 2.37), que entrega exactamente 5V, y no un adaptador de pared, ya que el voltaje de salida no siempre es el mismo del que indica su fabricante, por último se puede utilizar un circuito con un diodo zener de 5.1V.

3. No sobrepasar los niveles de corriente, tanto de entrada como de salida, el PIC puede entregar por cada uno de sus pines una corriente máxima de 25 mA. Asimismo soporta una corriente máxima de entrada de 25 mA, esto quiere decir que se puede encender un led con una resistencia de 330 Ω, como se observa en la Figura 2.38.

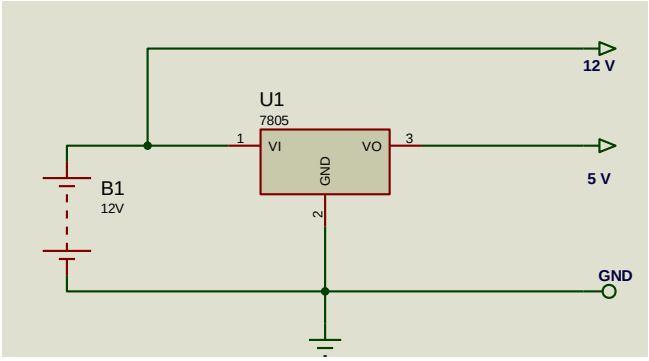


Figura 2. 37 Regulador de voltaje 7805.

Voltaje de salida del PIC= 5V.

Corriente que requiere el led para un encendido normal = 15 mA.

Entonces la resistencia necesaria para encender el led correctamente es:

$$V = RxI \qquad \text{Ec. 2.5}$$

$$R = \frac{V}{I} \qquad R = \frac{5V}{0.015A.} \qquad R = 333{,}33\Omega \approx 330\Omega$$

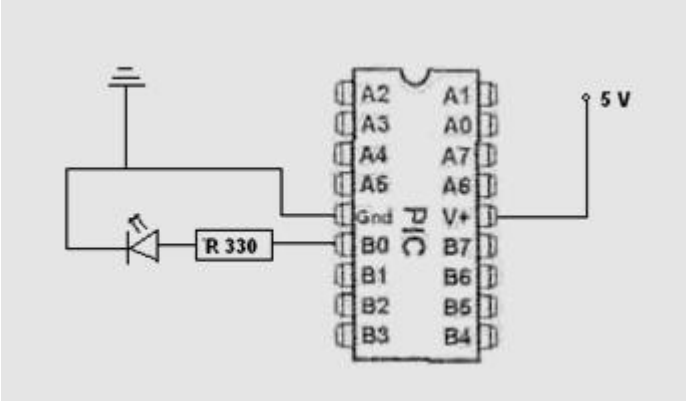


Figura 2. 38 Resistencia para un led.

Si se desea poner un pulsador en el PIC, la resistencia mínima a colocarse se calcula de la siguiente manera:

La corriente de entrada que soporta cada pin del PIC es de 25 mA, entonces para un pulsador se tiene.

$$R = \frac{V}{I} \qquad R = \frac{5V}{0.025A.} \qquad R = 333{,}33\Omega \quad \approx \quad 220\Omega$$

Esto quiere decir que la resistencia mínima a colocarse sería de 220Ω para estar al límite de la capacidad que soporta el PIC; pero no es muy aconsejable trabajar con los límites, por lo que se trabajará con una corriente de 1mA, teniendo el cálculo para este amperaje el siguiente (Figura 2.39).

$$R = \frac{V}{I} \qquad R = \frac{5V}{0.001A.} \qquad R = 5000\Omega \quad \approx \quad 4.7\text{K}\Omega$$

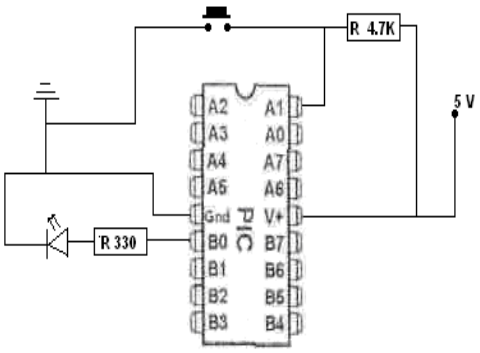


Figura 2. 39 Pulsante en un PIC.

4. Es necesario también colocar un capacitor de 1 uF en paralelo al PIC, éste evita malos funcionamientos que puede ocurrir, en especial cuando se utilizan teclados matriciales y se tiene conectado adicionalmente un buzzer activo (parlante activo o chicharra) y relés.

5. Cuando se necesite precisión en el PIC, se recomienda utilizar un cristal oscilador externo de 4 MHZ en adelante, ya que el oscilador interno que posee no tiene muy buena precisión.

Las anteriores especificaciones son las más básicas para montar un proyecto con PIC; dentro del diseño para el PLC, se tiene las siguientes especificaciones:

- Las entradas al PLC son a 10 Vdc, y como ya se anotó anteriormente el PIC solo soporta un máximo de 5 Vdc y 25mA, por lo que se realiza un divisor de voltaje.

- Se tiene una entrada de 10 Vdc y se necesita bajar a 5 V y se dispone una resistencia de 10KΩ (Figura 2.40).

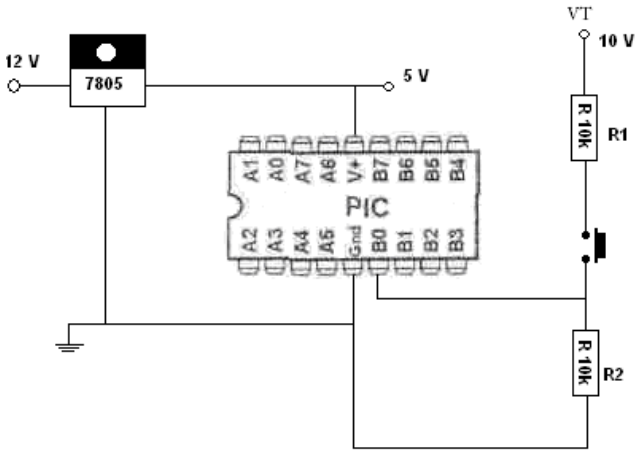


Figura 2. 40 Divisor de voltaje.

$$V1 = \frac{V_T x R_1}{R_{EQ}} = R_{EQ} \frac{V_T x R_1}{V_1} \qquad \text{Ec 2.6}$$

$$R_{EQ} = \frac{10x10000}{5} = 20000 = 20K\Omega$$

R_{EQ} =R1 + R2

R2 = R_{EQ} -R1

R2= 20 – 10 = 10 KΩ

La corriente que circulará por el PIC será:

$$I = \frac{V_T}{R_{EQ}} \qquad \text{De la Ec 2.5}$$

$$I = \frac{10}{20000} = 0.5mA$$

El cual es correcto ya que el PIC soporta 25mA en modo sumidero por cada pin, como se observa en la Figura 2.41.

El diagrama unifilar para las 16 entradas, y el diagrama para el diseño de la placa se observa en Anexos.

6. Para el diseño de las salidas, se sabe que el PIC entrega 5 Vdc y para que el proyecto funcione a 120 V, se utilizará periféricos de salida, como relès.

7. Se colocará un transistor 2N3904, el cálculo es el siguiente:

Datos del 2N3904

TBJ NPN 2N3904		
P_D máxima	625	mW
V_{BE} on	0,7	V
V_{CE} máxima	40	
V_{CE} saturac.	0,3	
I_C máxima	200	mA
I_B mínima	0,05	
β mínimo	100	-

Tabla 2. 5 Características del Transistor 2N3904.

$V_B = 0{,}7V$ con el que se activa el transistor, como se observa en la Tabla 2.5

$$I_B = \frac{Ic}{\beta}$$ De la Ec 2.1

$$I_B = \frac{200mA}{200}$$

$$I_B = 1mA$$

I_B = Corriente Base.

Ic = Corriente Colector.

β = Ganancia

Observando la Figura 2.41, la resistencia para activar al transistor seria:

$$R_2 = \frac{V_{pic} - V_B}{I_{R2}} \qquad \text{Ec 2.7}$$

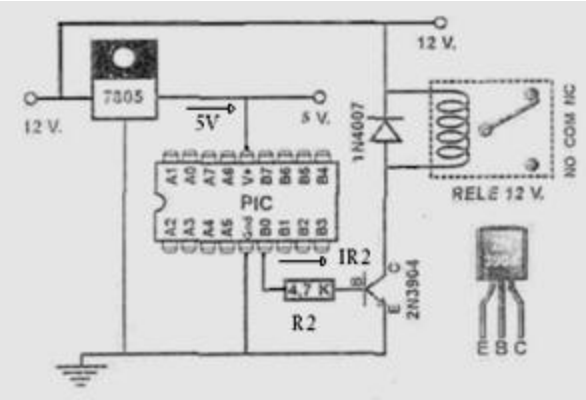


Figura 2. 41 Diagrama esquemático de conexión relé a un PIC.

Entonces:

$$R_{2\min} = \frac{5V - 0.7V}{0.0003A} = 1.4K\Omega$$

$$R_{2\max} = \frac{5V - 0.7V}{0.0005A} = 8.6K\Omega$$

Realizando una media entre los dos resultados se obtiene: $R_{eq2} \approx 4.7K\Omega$

Corriente de la bobina del relé.

$$I_{rele} = 1A$$

- También se coloca un diodo para la protección del microcontrolador, esto permite evitar que el relé vibre o se desconecta esporádicamente.

Corriente del diodo

$$I_D \geq 2I_{rele}$$

$$I_D \geq 2 \times 1A$$

$I_D \geq 2A$ Con esta corriente el diodo indicado es el diodo 1N4007.

8. Para la salida se utiliza salidas por relé, ya que se puede manejar cargas de corriente alterna y continua; y se tiene la ventaja que la separación entre la carga y los circuitos de control es altamente confiable, como se puede observar en la Figura 2.41.

Los relés a utilizar serán de 12Vdc a la bobina y soportar una carga de 220 Vac y 7 Amperios de corriente.

Para las entradas análogas, que son muy importantes en el campo de control de procesos, algunos PLC de baja gama carecen de entradas y salidas análogas, ya que su manejo es más complejo en comparación a las señales digitales.

Las entradas análogas deben ser transformadas en señales digitales, y a su vez las respuestas digitales que entrega el PIC deben ser transformadas a señales análogas, este proceso es denominado conversión análogo - digital y digital - análogo, respectivamente.

En el presente proyecto para las entradas análogas no se utiliza ninguna conversión ya que el voltaje de entrada es de 0 a 10 Vdc; y como el PIC soporta un voltaje de 5 Vdc, se diseñó un divisor de voltaje.

Para las salidas análogas se utiliza un conversor digital análogo utilizando un amplificador operacional, configurado como sumador que es el que se encarga de sumar los valores de cada una de las derivaciones; en la Figura 2.42 se observa el esquema interno de una salida analógica incluyendo conversor digital análogo.

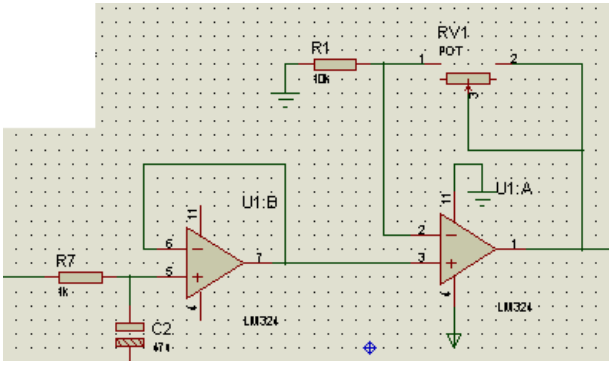


Figura 2. 42 Diagrama unifilar de salida digital análogo.

Se tiene a la salida del PIC un PWM, esto quiere decir, un tren de pulsos de 5V. La amplitud y frecuencia del ancho de pulso están previamente programados en el microcontrolador. La señal pasa por un filtro RC para obtener el valor medio del PWM de 5V a 2.3V, el cálculo para la resistencia y el capacitor están dados por la siguiente ecuación:

$$V_{Medio} = \frac{1}{2\pi RC} \qquad \text{Ec 2.8}$$

$$V_{Medio} = 2.3V$$
$$R = 1K$$
$$C = 47uF$$

Donde:

V_{Medio}= Voltaje medio al que se desea llegar.
R= Resistencia condicional, para poder calcular el valor del capacitor, si el valor del capacitor es alto, se deberá trabajar con el valor de R hasta obtener un valor del capacitor deseado, en este caso se escogió una resistencia de 1K, con el cual el valor del capacitor es 47 uF.

A continuación se utilizará dos amplificadores operacionales, el primer amplificador actúa como un buffer, y el segundo amplifica la señal; como se necesita una salida de 10V, se necesita un amplificador con ganancia 2.

El cálculo para las resistencias del amplificador está dado por la siguiente ecuación:

$$G = 1 + \frac{RV1}{R1} \qquad \text{Ec 2.9}$$

$$R1 = 10K$$

$$RV1 = 20K$$

$$G = 2$$

Donde:

R1= Resistencia condicionada, para el cálculo de RV1, según la ganancia que se requiera se podrá ir variando el valor de RV1.

De esta manera se concluye con el cálculo de los elementos para el diseño y construcción de las placas del PLC.

La fotografía de placa terminada se indica en Anexos.

- **Optoacoplador 4N35**

Son conocidos como optoaisladores o dispositivos de acoplamiento óptico, basan su funcionamiento en el empleo de un haz de radiación luminosa para pasar señales de un circuito a otro sin conexión eléctrica. Éstos son muy útiles cuando se utilizan por ejemplo, Microcontroladores PICs, para proteger el microcontrolador este dispositivo es una buena opción. En general pueden sustituir a los relés ya que tienen una velocidad de conmutación mayor, así como, la ausencia de rebotes, un circuito típico con optoacoplador se muestra en la Figura 2.43.

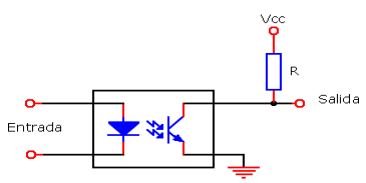


Figura 2. 43 Circuito con optoacoplador.

La gran ventaja de un optoacoplador reside en el aislamiento eléctrico que puede establecerse entre los circuitos de entrada y salida. Fundamentalmente este dispositivo está formado por una fuente emisora de luz, y un fotosensor de silicio, que se adapta a la sensibilidad espectral del emisor luminoso, todos estos elementos se encuentran dentro de un encapsulado (Figura 2.44).

Figura 2. 44 Optoacopladores encapsulados.

El optoacoplador que se utiliza para el PLC es de tipo fototransistor, que se compone de un diodo IR con una etapa de salida formada por un transistor BJT (Figura 2.45).

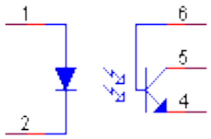


Figura 2. 45 Optoaclopador tipo Fototransistor

En la Figura 2.46 se puede apreciar el circuito esquemático de conexión del optocoplador con el relé.

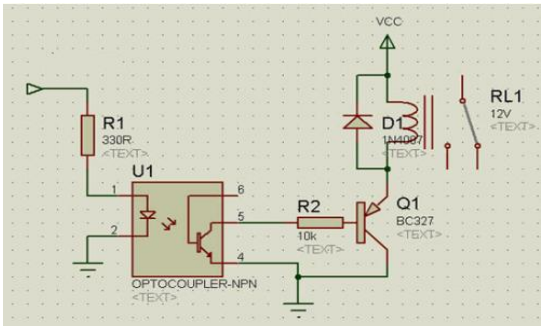


Figura 2. 46 Circuito esquemático del optoacoplador con el relé.

En la Tabla 2.6 se observa las principales características técnicas del optoacoplador 4N35.

PARAMETER		TEST CONDITIONS			MIN	TYP	MAX	UNIT
$V_{(BR)CBO}$	Collector-base breakdown voltage	I_C = 100 µA,	I_E = 0,	I_F = 0	70†			V
$V_{(BR)CEO}$	Collector-emitter breakdown voltage	I_C = 10 mA,	I_B = 0,	I_F = 0	30†			V
$V_{(BR)EBO}$	Emitter-base breakdown voltage	I_E = 100 µA,	I_C = 0,	I_F = 0	7†			V
I_R	Input diode static reverse current	V_R = 6 V					10†	µA
I_{IO}	Input-to-output current	V_{IO} = rated peak value,		t = 8 ms			100	mA
$I_{C(on)}$	On-state collector current	V_{CE} = 10 V,	I_F = 10 mA,	I_B = 0	10†			mA
		V_{CE} = 10 V, T_A = −55°C	I_F = 10 mA,	I_B = 0,	4†			
		V_{CE} = 10 V, T_A = 100°C	I_F = 10 mA,	I_B = 0,	4†			
$I_{C(off)}$	Off-state collector current	V_{CE} = 10 V,	I_F = 0	I_B = 0		1	50	nA
		V_{CE} = 30 V, T_A = 100°C	I_F = 0,	I_B = 0,			500†	µA
h_{FE}	Transistor static forward current transfer ratio	V_{CE} = 5 V,	I_C = 10 mA,	I_F = 0		500		
V_F	Input diode static forward voltage	I_F = 10 mA			0.8†		1.5†	V
		I_F = 10 mA,	T_A = −55°C		0.9†		1.7†	
		I_F = 10 mA,	T_A = 100°C		0.7†		1.4†	
$V_{CE(sat)}$	Collector-emitter saturation voltage	I_C = 0.5 mA,	I_F = 10 mA,	I_B = 0 mA			0.3†	V
r_{IO}	Input-to-output internal resistance	V_{IO} = 500 V,	See Note 6		10^{11}†			Ω
C_{io}	Input-to-output capacitance	V_{IO} = 0,	f = 1 MHz,	See Note 6		1	2.5†	pF

Tabla 2. 6 Características del optoacoplador 4N35.

- **Display LCD[25].**

Una pantalla de cristal líquido o LCD (Liquid crystal display) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.

Cada píxel de un LCD típicamente consiste de una capa de moléculas de un cristal líquido especial colocadas entre dos electrodos de vidrio y dos filtros de polarización. Los filtros polarizadores se colocan con sus ejes perpendiculares entre sí. Sin cristal líquido entre el filtro polarizador, la luz que pasa por el primer filtro sería completamente bloqueada por el segundo polarizador, ya que éste se halla con su eje óptico rotado 90º respecto al primero.

Antes de la aplicación de un campo eléctrico sobre los electrodos de vidrio, la orientación de las moléculas de cristal líquido está determinada por unas micro-muescas que se hacen en las superficies de los vidrios. Típicamente las direcciones de alineación de la superficie de los dos electrodos son perpendiculares entre sí, y así se organizan las moléculas en una estructura helicoidal, o retorcida. La luz que pasa a través del primer filtro polarizador va

[25] http://www.dalsa.com/shared/content/Photonics_Spectra_CCDvsCMOS_Litwiller.pdf

cambiando su polarización de acuerdo con la hélice de cristal líquido como se muestra en la Figura 2.47, y de este modo logra pasar por el segundo filtro polarizador.

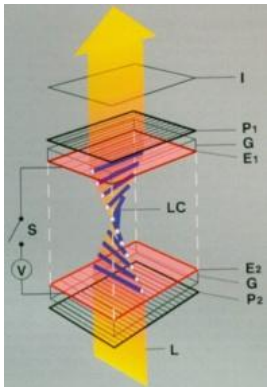


Figura 2. 47 Luz a través del filtro polarizador.

Cuando se aplica un voltaje a través de los electrodos, una fuerza de giro orienta las moléculas de cristal líquido paralelas al campo eléctrico. Es decir, se distorsiona la estructura helicoidal. De este modo se reduce la rotación de la polarización de la luz incidente, y la luz ya no logra atravesar el segundo polarizador. Si la tensión aplicada es lo suficientemente grande, las moléculas de cristal líquido en el centro de la capa son casi completamente desenrolladas y la polarización de la luz incidente no es rotada por el cristal líquido. Esta luz será bloqueada por el segundo filtro polarizador y el pixel aparecerá negro (Figura 2.48). Por el control de la tensión aplicada a través de la capa de cristal líquido en cada píxel, la luz se puede permitir pasar a través de distintas cantidades, constituyéndose los diferentes tonos de gris.

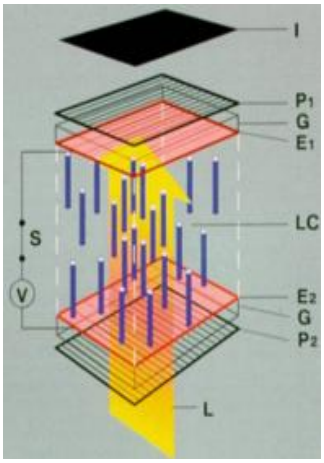


Figura 2. 48 Luz bloqueada por el segundo filtro.

El módulo LCD con el que se trabajó tiene 20 pines, como se observa en la Tabla 2.7 de distribución de pines .Su alimentación es de +5 V. Para el módulo de 8 bits se requiere 11 líneas (uno de 4 bits, necesitaría sólo 7). De ellas hay tres de control, que son EN (habilitación), I/D (Instrucción/Datos) y R/W (Lectura/Escritura).
En un modo de 4 bits se usarían sólo las líneas DB4-DB7 de datos.

Pin No.	Symbol	Level	Description
1	Vdd	5.0V	Supply voltage for logic and LCD (+)
2	Vss	0V	Ground
3	V0	-	Operating voltage for LCD (variable)
4~11	DB0~DB7	H/L	Data bit 0~7
12	CS2	L	Chip select signal for IC2
13	CS1	L	Chip select signal for IC1
14	/RES	L	Reset signal
15	R/W	H/L	H: read (MUP<－module),L: write (MPU－>module)
16	D/I	H/L	H: data, L: instruction code
17	E	H, H L	Chip enable signal
18	VEE	-	Operating voltage for LCD (variable)
19	A	4.2V	Backlight power supply
20	K	0V	Backlight power supply

Tabla 2. 7 Distribución de pines del LCD.

Las características eléctricas del LCD 128 x 64 se detallan a continuación en la Tabla 2.8.

(VDD= 4.5 to 5.5V, VSS=0V,VDD-VEE=8~17V,Ta= -30 to +85℃)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit	Note
Operating voltage	V_{DD}	-	4.5	-	5.5	V	
Input voltage	V_{IH}	-	$0.7V_{DD}$	-	V_{DD}	V	*1
	V_{IL}	-	V_{SS}	-	$0.3V_{DD}$	V	*1
output voltage	V_{OH}	I_{OH}= -0.4mA	V_{DD}-0.4	-	-	V	*2
	V_{OL}	I_{OL}= 0.4mA	-	-	0.4	V	*2
Input leakage current	I_{LKG}	V_{IN}= V_{DD} ~ V_{SS}	-1.0	-	+1.0	μA	*1
OSC Frequency	fosc	Rf=47kΩ±2% Cf=20pF±5%	315	450	585	kHz	
On Resistance (Vdiv-Ci)	R_{ONS}	V_{DD}-V_{EE}=17V Load current±150μA	-	-	1.5	kΩ	
Operating current	I_{DD1}	Master mode 1/128 Duty	-	-	1.0	mA	*3
	I_{DD2}	Master mode 1/128 Duty	-	-	0.2	mA	*4
Supply Current	I_{EE}	Master mode 1/128 Duty	-	-	0.1	mA	*5
Operating	f_{op1}	Master mode External Duty	50	-	600	kHz	
Frequency	f_{op2}	Slave mode	0.5	-	1500	kHz	

Tabla 2. 8 Características eléctricas.

Las conexiones del LCD gráfico al microcontrolador se observan en la Figura 2.49.

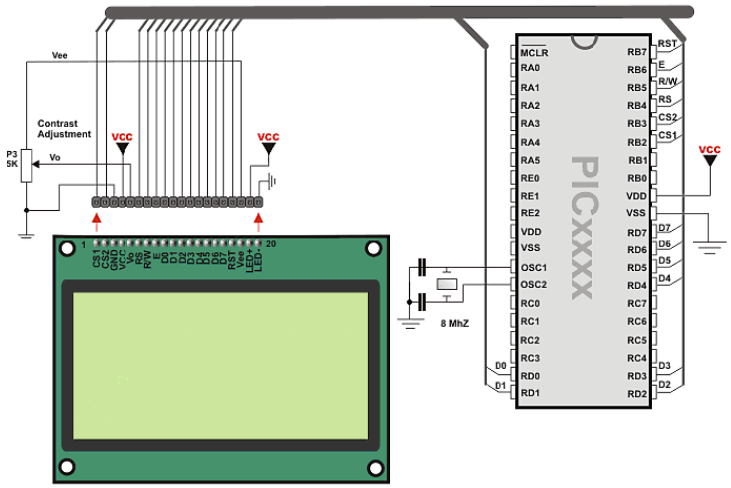


Figura 2. 49 Conexión del LCD.

2.7 CONSTRUCCIÓN DE UNA PCB.[26]

A continuación se indican los pasos para realizar una placa electrónica.

Primero se comenzará por describir los materiales a utilizarse:

Placa de cobre: Puede ser cualquiera de Pertinax o Fibra de vidrio. La de fibra de vidrio es más costosa, pero tiene propiedades mecánicas mejores que el pertinax, normalmente de un color blanco frente al café opaco del pertinax.

Acido para placas (Cloruro Férrico): Lo venden en distintas cantidades, desde medio litro hasta cinco litros o más. Medio litro debería ser más que suficiente para hacer unas 20 placas (de tamaño pequeño, no una placa madre), sobretodo sabiendo que una dosis de ácido puede usarse dos o tres veces, aunque se va demorando cada vez más en eliminar el cobre de la placa.

Plancha: Para realizar la transferencia de calor.

[26] Departamento de Ingeniería Eléctrica Universidad de Chile.

Recipiente de Plástico: Como el ácido es irritante de la piel, Ideal que sea de tamaño reducido (lo más ajustado posible a la placa) y plano de manera de ahorrar ácido.

Papel para Transferencia: Depende del método a usar, el de las transparencias o el del papel fotográfico. Se debe tener cuidado en que se imprima bien el circuito y que el toner se pueda desprender al rasparlo con la uña.

Lija Fina: Cualquier lija muy fina (de alto grano) como las que se utilizan para metales. En su defecto se puede utilizar una escobilla metálica.

Agente Limpiador: Lo que se utiliza normalmente en circuitos, ya que posee la cualidad de ser un buen limpiador y no conducir la electricidad. También se puede usar algún tipo de abrasivo, como por ejemplo la acetona.

Lápiz Permanente: Bastante útil a la hora de reparar errores, en general cualquier lápiz que sea permanente.

2.7.1 Instrucciones para la realización de la PCB.

1) Para comenzar, se debe traspasar el diseño del circuito (PCB) al papel para transferencia. Lo mejor es hacerlo con una impresora láser, si es que se posee el diseño de forma electrónica (y se dispone de una impresora láser). Se debe imprimir con la máxima resolución posible de manera que quede la mayor cantidad de toner sobre las pistas, como se muestra en la Figura 2.50, incluso se puede imprimir dos veces, una encima de otra, si se tiene un buen sistema para alinear las hojas.

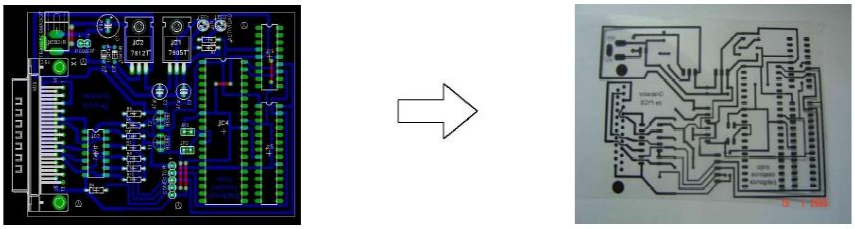


Figura 2. 50 Impresión del PCB.

2) Con el diseño ya impreso en el papel especial, preparar la placa de cobre. Para la mejor transferencia del toner, lijar la placa de cobre con la lija fina de manera de limpiarla y dejarla rayada, como se aprecia en la Figura 2.51.

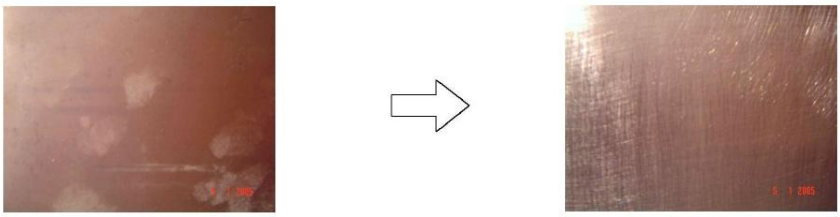


Figura 2. 51 Lijado de placas.

3) Para la transferencia del circuito a la placa de cobre, en primer lugar se debe calentar la plancha lo más caliente y sin vapor. Luego, se coloca la transparencia con el circuito impreso sobre la placa de cobre (Figura 2.52), asegurándola con un poco de cinta adhesiva.

Figura 2. 52 Circuito impreso a la placa de cobre.

A continuación, se coloca una hoja blanca sobre el reverso de la transparencia adherida a la placa, de manera de no aplicar la plancha directamente sobre el plástico de la transparencia. Con el calor de la plancha, el toner deberá empezar a pegarse a la placa en un par de minutos como se observa en la Figura 2.53.

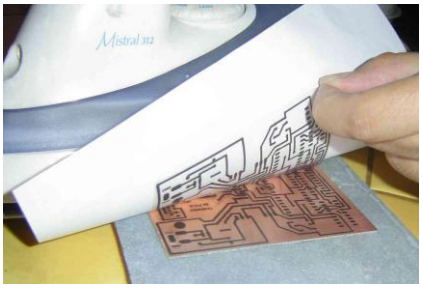


Figura 2. 53 Impresión de la placa de cobre.

Si se siguen los pasos anteriores correctamente, el toner del papel de transferencia quedará pegado ahora en la placa, formando las líneas y pistas del circuito final (Figura 2.54). Normalmente en esta etapa quedan algunas líneas borrosas (sobretodo en alguna esquina), lo cual puede ser corregido usando un lápiz permanente, preferentemente de punta fina.

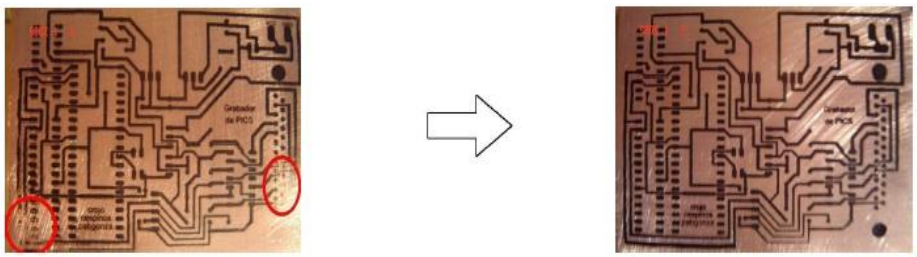


Figura 2. 54 Impresión de pistas en la placa.

4) Ya con el circuito impreso en la placa de cobre, se procede a:
En primer lugar se debe poner la placa en un recipiente de plástico, y comenzar a colocar el ácido con cuidado hasta tapar la placa. Luego se debe esperar que el ácido diluya todo el cobre excepto el cubierto por el toner (Figura 2.55). El proceso es relativamente lento (como 20 minutos) pero se puede acelerar agitando el recipiente o poniéndolo a baño maría.

Figura 2. 55 Placa en recipiente, con el ácido.

Cuando se observe que no quedan restos de ácido en los sectores libres de toner, se puede extraer la placa del recipiente. Se debe lavar con abundante agua para eliminar los restos del ácido. En la Figura 2.56 se aprecian los resultados de esta etapa.

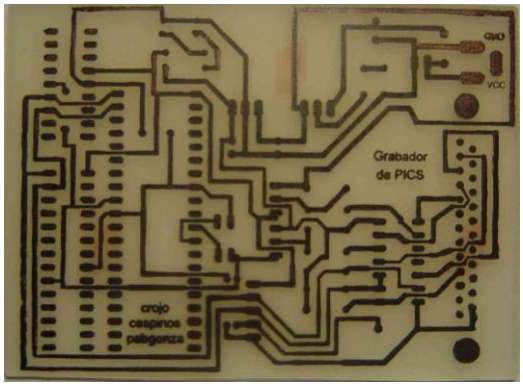


Figura 2. 56 Placa una vez sacada del recipiente.

Por último lo que queda es limpiar la placa, perforar con un taladro con una broca adecuada, y soldar los elementos que contenga dicha placa, como se observa en la Figura 2.57.

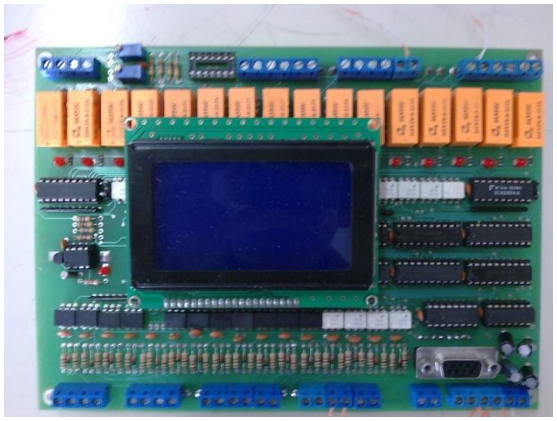


Figura 2. 57 Placa una vez finalizada.

Una vez finalizada la placa, se procede a armar la carcasa del PLC, como se muestra en la Figura 2.58 (Anexo 1).

Figura 2. 58 PLC ByL

CAPÍTULO III

3. IMPLEMENTACIÓN Y MANEJO DEL SOFTWARE EN VISUAL BASIC

3.1 INTRODUCCIÓN A VISUAL BASIC.

El lenguaje de programación BASIC (Beginner's All purpose Symbolic Instruction Code) nació en el año 1964 como una herramienta destinado a principiantes, buscando una forma sencilla de realizar programas, empleando un lenguaje casi igual al usado en la vida ordinaria (en inglés), y con instrucciones muy sencillas y escasas. Teniendo en cuenta el año de su nacimiento, este lenguaje cubría casi todas las necesidades para la ejecución de programas. Téngase en cuenta que las máquinas existentes en aquella época estaban estrenando los transistores como elementos de conmutación, los ciclos de trabajo llegaban a la impensable cifra de 10.000 por segundo.

La evolución del BASIC por los años 70 fue escasa, dado el auge que tomaron en aquella época lenguajes de alto nivel como el FORTRAN y el COBOL. En 1978 se definió una norma para unificar los Basics existentes creándose la normativa BASIC STANDARD

Con la aparición de los primeros ordenadores personales, dedicados comercialmente al usuario particular, allá por la primera mitad de los ochenta, el BASIC resurgió como lenguaje de programación pensado para principiantes, y muchos de estos pequeños ordenadores lo usaban como único sistema operativo.

Los programadores profesionales no llegaron a utilizarlo, habida cuenta de las desventajas de este lenguaje respecto a otras herramientas (PASCAL, C, CLIPPER). El BASIC con estas versiones para PC llegó incluso a perder crédito entre los profesionales de la informática.

Las razones para ello eran obvias:

- No era un lenguaje estructurado.
- No existían herramientas de compilación fiables.

- No disponía de herramientas de intercambio de información.
- No tenía librerías.
- No se podía acceder al interior de la máquina.

Tal fue ese abandono por parte de los usuarios, que la aparición del Quick-BASIC (Q-Basic) de Microsoft, una versión ya potente del BASIC, que corregía casi todos los defectos de las versiones pasó prácticamente inadvertida.

Sin embargo algo había en el BASIC que tentaba a superarse: su gran sencillez de manejo. Si a esto se le añade el entorno gráfico Windows, el aprovechamiento al máximo de las posibilidades de Windows en cuanto a intercambio de información, de sus librerías, de sus drivers y controladores, manejo de bases de datos, etc. el producto resultante puede ser algo que satisfaga todas las necesidades de programación en el entorno Windows. La suma de todos estos aspectos es VISUAL BASIC (VB). Esta herramienta conserva del BASIC de los años 80 únicamente su nombre y su sencillez, y tras su lanzamiento al mercado, la aceptación a nivel profesional hizo borrar por fin el "mal nombre" asociado a la palabra BASIC.

Es obligado decir sin embargo, que sigue siendo BASIC un software de un alto nivel de programación. No se pueden comparar sus prestaciones con otros lenguajes cuando se desea llegar al fondo de la máquina y controlar uno a uno sus registros. No es ese el fin perseguido con VB y si es necesario llegar a esas precisiones será necesario utilizar otro lenguaje que permita bajar el nivel de programación, o realizar librerías (DLLs) que lo hagan. En la mayor parte de las aplicaciones, las herramientas aportadas por VB son más que suficiente para lograr un programa fácil de realizar y de altas prestaciones.

3.1.1 Características Generales de Visual Basic.

Visual Basic es una herramienta de diseño de aplicaciones para Windows, en la que éstas se desarrollan en una gran parte a partir del diseño de una interface gráfica. En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interface gráfica.
Es por tanto un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias. Ya que no se puede decir que VB pertenezca por completo a uno de esos dos

tipos de programación, se debe inventar una palabra que la defina: PROGRAMACIÓN VISUAL.

La creación de un programa bajo Visual Basic lleva los siguientes pasos:

- **Análisis:** Es el estudio de las necesidades que han dado origen a la creación de ese programa. Es lo que se llama *Análisis* de la aplicación. Es la primera fase que debe tener siempre un programa y es también la más olvidada entre los programadores. Una aplicación no se inicia con el teclado, sino sobre un papel.

- **Creación de un interface de usuario**: Este interface será la principal vía de comunicación hombre-máquina, tanto para salida de datos como para entrada. Será necesario partir de una o varias ventanas - Formularios - a las que se irá añadiendo los controles necesarios.

- **Definición de las propiedades de los controles:** Se dará la forma, posición, y todas las características necesarias a los controles que se haya colocado en ese formulario. Estas propiedades determinarán la forma estática de los controles; es decir, como son los controles y para qué sirven.

- **Generación del código asociado a los eventos que ocurran a estos controles:** A la respuesta a estos eventos (clic, doble clic, una tecla pulsada, etc.) se le llama Procedimiento, y deberá generarse de acuerdo a las necesidades del programa.

- **Generación del código del programa**: Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Sin embargo, VB ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados Funciones, y otros llamados Procedimientos. Estos Procedimientos no responden a un evento acaecido a un control o formulario, sino que responden a un evento producido durante la ejecución del programa.

3.1.2 Variables Definición.

Basic, desde siempre, al contrario de otros sistemas de programación, no exigió la definición previa de una variable. Una variable, como debe seguro conocer, es un nombre que en el programa se le asigna a un dato. Ese dato podrá cambiar.

3.1.3 Tipos de Variables.

Las variables pueden ser de los siguientes tipos: (El número indicado en la segunda columna indica el número de Bytes que ocupa en memoria.)

Booleana	(2)	Admite los valores 0 y 1, o True (verdadero) y False (falso)
Byte	(1)	Números enteros, en el rango de 0 a 255
Integer	(2)	Números enteros en el rango de -32768 a 32767
Long	(4)	Números enteros en el rango de -2147483648 a 2147483647
Single	(4)	Punto flotante, simple precisión
Doble	(8)	Punto flotante, doble precisión.
Currency	(8)	Entero, con punto decimal fijo (Típico de monedas)
String	(*)	Cadenas alfanuméricas de longitud variable o fija
Date	(8)	Fechas
Objet	(4)	Referencia a objetos
Variant	(**)	Otros tipos de datos

(*) Una variable tipo String ocupa el mismo número de bytes que caracteres tenga la cadena.
(**) Una variable tipo Variant ocupa 16 bytes si se trata de un número, y 22 bytes + longitud de la cadena si se trata de un dato tipo cadena de caracteres.
Existen también variables definidas por el usuario. En este tipo de variables se pueden introducir muchos datos de distinto tipo. Los bytes necesarios para almacenar esa variable dependerán de los datos que se hayan definido.
Dentro de las variables Object (variables que se refieren a objetos) existe una gran variedad, como las variables que pueden ser del tipo Form - Formulario - , tipo Recordset, etc.

Cada tipo de variable ocupa unos determinados bytes. Si no se define una variable, VB toma como tipo por defecto para la variable el tipo Variant. Este tipo ocupa más bytes que, por ejemplo, un integer. Si el tipo de dato que se va a introducir en una variable es un integer, y no la se ha declarado como tal, VB asumirá para esa variable que es del tipo Variant, lo que le llevará a gastar más bytes de memoria (16) que los que necesitaría (2), si se hubiese declarado previamente. Si esa variable va a estar en el rango de 0 a 255, y no se

ha declarado previamente que la variable va a ser del tipo Byte, o se la declara como integer, y se está desperdiciando memoria RAM y posiblemente, retardando la ejecución del programa. Lo mismo se puede decir del resto de las variables, y lo importante es declararlas y declararlas bien.

Puede declarar el tipo de la variable mediante un carácter después del nombre de la variable. Esta técnica, obligatoria en Quick-Basic, está en desuso en VB. No es recomendable definir el tipo de esta forma, pues existe un serio peligro de error. De cualquier forma, eso es libre del programador y de sus costumbres.

Los caracteres que definen cada tipo de variable son:

%	Integer	&	Long	!	Single
#	Double	@	Currency	$	String

3.1.4 Declaración de Variables.

- **Sentencia DIM .** Es la forma más común de declarar una variable como Privada. Puede emplearse en un Procedimiento, Función, Formulario o Módulo. La sintaxis es de la siguiente forma:

Dim nombrevariable **As Integer** (o el tipo que sea)

Declarando una variable con la sentencia DIM, en un *formulario, Función, procedimiento* o *módulo*, el ámbito de la variable será el explicado posteriormente para una variable declarada como Privada. Es decir, esa variable no sale del formulario, procedimiento ó módulo donde se declaró. Cada vez que ingrese al formulario, procedimiento o módulo, esa variable tomará el valor cero (si es numérica) o nulo (si es string).

- **Sentencia PRIVATE.** Es la forma de declarar una variable como Privada. Puede emplearse solamente en la sección de declaraciones de un Formulario o Módulo. La sintaxis es de la siguiente forma:

Private nombrevariable **As** Tipovariable

Declarando una variable mediante la sentencia **PRIVATE** en un Formulario o Módulo, esa variable puede usarse en todo ese Formulario o Módulo (En todos sus Procedimientos y Funciones), pero NO fuera del Formulario o Módulo donde se declaró. La sentencia **Private** no puede usarse en un procedimiento o función.

- **Sentencia PUBLIC.** Es la forma de declarar una variable como Pública. Puede emplearse solamente en la sección de declaraciones de un Formulario o Módulo. La sintaxis es de la siguiente forma:

Public nombrevariable **As** Tipovariable

Declarando una variable de esta forma en la sección de declaraciones de un **Módulo**, esa variable puede usarse en cualquier parte del programa citándola simplemente por su nombre.
Si se declara de esta forma en la sección de declaraciones de un Formulario, esa variable puede usarse en todo el programa. Para nombrarla, en el Formulario donde se declaró basta con citarla por su nombre. Si no está en ese Formulario, habrá que citarla por el nombre del Formulario, seguido del nombre de la variable, separados por un punto:

NombreFormulario.Nombrevariable

En un Módulo puede usarse también la sentencia **Global** en vez de **Public**:

- **Sentencia GLOBAL.** Declara una variable que es válida en todo el programa. La sintaxis es:

Global nombrevariable **As** tipovariable

La sentencia **Global** sólo puede usarse en el apartado de declaraciones de un **Módulo**.

- **Sentencia STATIC.** Como se dijo anteriormente, una variable declarada en un procedimiento pierde su valor al salir de él. Lo peor es que una vez que el programa vuelva a entrar en ese procedimiento, la variable estará puesta a cero. Afortunadamente, esto último tiene solución. Si se declara una variable en un

procedimiento o función, como estática; esa variable, aunque no se podrá utilizar fuera de ese procedimiento o función, cuando se vuelva a él conservará el valor que tenía cuando lo cerró. Esta declaración como estática se realiza mediante la instrucción **Static.**

Static nombrevariable **As** tipovariable

El nombre de una variable puede ser tan largo como se desee, hasta un máximo de 40 caracteres. Es indiferente usar mayúscula ó minúsculas. El nombre de una variable siempre debe comenzar por una letra.

Al compilar el programa no se lleva el nombre, es decir, no va a ocupar más espacio. Se debe utilizar nombres que definan la variable con algún sentido.

Pese a que Visual Basic no obliga a declarar variables, es muy útil hacerlo. De esta forma se tiene control sobre el programa.

3.1.5 Sentencias Condicionales.

Se llaman sentencias condicionales a aquellas que se realizan si se cumple una determinada condición. La sentencia condicional más usada es:

- **IF – THEN**

Si se cumple una condición **Entonces**
Realiza estas instrucciones
Si no se cumple
Realiza estas otras instrucciones
Fin de la sentencia.

Lo descrito anteriormente toma la forma:

If condición **Then**
Instrucciones
Else
Otras instrucciones
End If

En este ejemplo, o se cumple una condición y ejecuta unas determinadas instrucciones, o no se cumple, y ejecuta otras instrucciones distintas. Puede ocurrir que en caso de no cumplirse la condición primera, se abra un abanico de dos o tres posibilidades. La sentencia condicional tendría entonces la forma:

If condición 1 **Then**
Instrucciones
ElseIf Condición 2
Otras instrucciones
ElseIf Condición 3
Otro juego de instrucciones
Else
Instrucciones que debe realizar caso de no cumplir las condiciones 1, 2 y 3.
End If

Existe otra sentencia condicional muy usada:

- **SELECT CASE.**

Su nombre casi define lo que es: Selecciona, dependiendo del caso, un determinado juego de instrucciones:

Select Case variable
En donde variable es una versátil que puede tomar los valores
Case 1
Instrucciones a ejecutar en caso de que variable = 1
Case 2
Instrucciones a ejecutar en caso de que variable = 2
Case 3
Instrucciones a ejecutar en caso de que variable = 3
Case 4
Instrucciones a ejecutar en caso de que variable = 4
Case Else
Instrucciones a ejecutar en caso de que variable sea distinta a los valores anteriores
End Select

Este procedimiento resulta mucho más sencillo y rápido que la sentencia **If Then Else;** cuando el margen de elección es mayor que 2.
Puede agrupar varios valores en una misma línea. Por ejemplo, si se tiene 100 valores posibles, y el tratamiento es igual para varios de ellos, se puede agruparlos así:

```
Select Case Valor
    Case Is < 20
        Instrucciones
    Case Is = 20
        Instrucciones
    Case Is = 21, 22, 23, 24, 25, 26
        Instrucciones
    Case Is > 26
        Instrucciones
End Select
```

3.1.6 Sentencias de Bucles.

Es muy común utilizar bucles a lo largo de un programa. Un bucle es una sucesión repetitiva de instrucciones, que se estarán realizando mientras se cumpla una condición o mientras no se cumpla otra condición:

Mientras condición
Instrucciones
Fin del bucle

Existen dos formas de bucle: Una, que realiza un número determinado de recorridos por el bucle; es el denominado bucle por contador. Otra, realiza el bucle hasta que se cumpla (o deje de cumplirse) una condición; es el llamado bucle por condición.

- Bucle por Contador.

Realiza el bucle tantas veces como se lo indique. Por ejemplo, en este bucle va a presentar las 26 letras mayúsculas del alfabeto.

```
For N=65 To 90
Label1.caption = Chr ( N )
Next N
```

Este "programa" se presenta en una caja (Label) los caracteres cuyo número ASCII vaya desde el 65 (A) al 90 (Z) Comenzará presentando el correspondiente al número 65, e irá presentando sucesivamente el 66, el 67, etc., hasta llegar al 90, donde se parará.

- Bucles por Condición.

Ejecuta las instrucciones del bucle mientras se cumple una condición

```
X = 0
Do While X < 1000
X = X + 1
Loop
```

El programa toma una variable (X) que previamente se colocó a cero, e incrementa su valor una unidad. Analiza si el valor de X es menor que 1000, y si es cierto, vuelve a realizar el bucle. Así hasta que X ya no sea menor que 1000. Al dejar de cumplirse que X sea menor que 1000, sale del bucle.

Si lo que se requiere es que el programa se ejecute mientras no se cumpla una determinada condición, la sentencia será:

```
X = 0
Do Until X > 1000
X = X + 1
Loop
```

Se observa que la diferencia entre una y otra es la condición, **While** para indicar **Mientras se cumpla que ...** y **Until** para indicar **Mientras no se cumpla que ....**

3.1.7 Formas de Salir de un Bucle.

Para salir de un bucle, se puede esperar a que se complete, bien porque se ha cumplido la condición (Bucles por condición) o a que se haya llegado al final de la cuenta (Bucles por contador). Pero puede ocurrir que le interese salir del bucle sin haber terminado la ejecución total del mismo, por ejemplo, porque ya se ha encontrado lo que se estaba buscando, y sería una pérdida de tiempo seguir ejecutando el bucle. Para ello, se debe ejecutar la sentencia **Exit Do**

En el siguiente ejemplo se pretende parar (stop) el programa cierto tiempo, a la espera que otro equipo se conecte a través del puerto serie. (Cuando se conecte, el control MSComm1 detectará la señal DSR que le envía el otro equipo y deberá salir inmediatamente del bucle para, por ejemplo, atender a ese puerto serie)

```
Contador = 0
Do While Contador < 1000
Contador = Contador + 1
DoEvents
If MSComm1.DSRHolding = True Then Exit Do
Sleep (1000)
Loop
```

A continuación se analiza en forma rápida la comunicación del puerto serie, ya que más adelante se explica más detenidamente el mismo.

En el siguiente ejemplo, se desea transmitir por el puerto serie una serie de números, del 1 al 1000, pero siempre y cuando que el equipo que está al otro lado de la línea serie esté a la escucha (Condición MSComm1.DSRHolding = True) Si se desconecta ese equipo, y la condición pasa de True a False, se desearía salir del bucle:

```
For I = 1 To 1000
If MSComm1.DSRHolding = False Then Exit Do
MSComm1.Output = Str(I)
Next I
```

- Función DoEvents.

Si se habla de bucles, necesariamente se debe hablar de la función DoEvents. Y es así por la misma razón que cada vez que se pone un bucle, debería poner dentro de ese bucle la palabra DoEvents.

Sí Visual Basic está ejecutando este código:

```
Do While Valor < 100
Valor = valor + 1
If Valor = 90 then Valor =1
Loop
```

Se observa que la variable "Valor" nunca podrá llegar a tener el valor 100, puesto que cada vez que llega a 90 se pone de nuevo el valor 1. El programa no va a salir nunca del bucle, pero además no podrá atender a otros eventos que se produzcan en el ordenador (por ejemplo la pulsación de una tecla) ya que el microprocesador está atendiendo solamente a ese bucle. Si lo está ejecutando desde el propio entorno de Visual Basic, deberá parar la ejecución mediante las teclas Ctrl – Pausa. No podrá pararlo mediante el botón Terminar de la barra de herramientas de VB. En un programa compilado posiblemente se tenga que reiniciar el equipo mediante **Alt – Ctrl – Sup**, llevándose por delante el resto de los programas que se estén ejecutando en ese momento en un PC.

Esto es lo que se llama un bucle infinito. Para que no suceda; o al menos, que si un programa entra en un bucle infinito, se pueda salir de una forma no tan agresiva, hay que introducir dentro del bucle la instrucción DoEvents. Cuando VB encuentra esta función el control del microprocesador pasa al sistema operativo, y si hay alguna operación en espera (la pulsación de una tecla, por ejemplo) la ejecuta y posteriormente vuelve a ejecutar el bucle. De esta forma, se puede salvar al menos los datos de las otras aplicaciones que se están ejecutando al mismo tiempo.

La programación del bucle será de la siguiente manera:

```
Do While Valor < 100
```

```
Valor = valor + 1
If Valor = 90 then Valor =1
DoEvents
Loop
```

3.1.8 Programación en Visual Basic.

Al entrar en el programa VB, aparecerá la pantalla de desarrollo en Visual Basic indicado en la Figura 3.1.

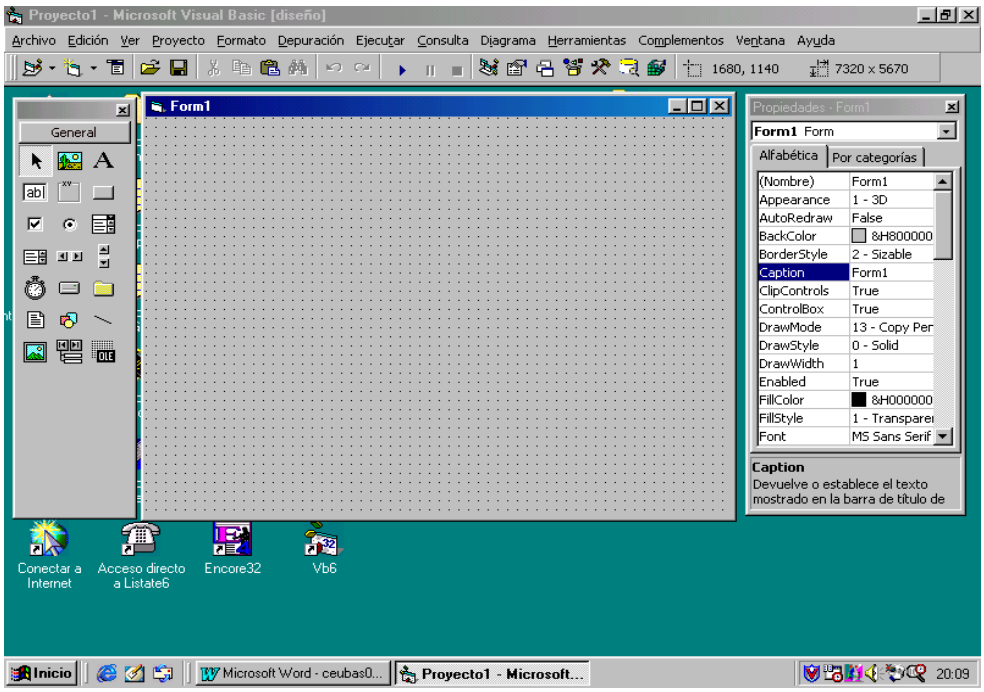


Figura 3. 1 Aspecto del entorno de desarrollo de aplicaciones VB.

En la pantalla anterior existen varios objetos. En la parte superior, la **barra de título** del programa Visual Basic, con el texto:

Proyect1 - Microsoft Visual Basic (Diseño)

Por debajo de esta barra de Título, se encuentra la **barra de menú** de VB, con las leyendas:

Archivo Edición Ver Proyecto Formato Depuración…. Etc.

Por debajo de esta barra de menú, se tiene la **barra de herramientas**, donde se podrá ver varios iconos, cada uno representa una determinada operación que puede realizar. Estas

operaciones están todas en la línea de menú, y puede acceder a ellas abriendo los menús desplegables que existen en cada una de las palabras Archivo Edición Ver..... ,de esta línea de menú.

A la izquierda o derecha de la pantalla se tiene una caja rectangular con varias columnas de iconos, conocida como **Caja de Herramientas.**

Posiblemente esta caja de herramientas no aparezca tal y como la ve en la Figura 3.1. Eso depende de la versión y la personalización. Para añadir un nuevo control, se debe hacer clic en **Proyecto Componentes** y aparecerá una caja con todos los controles existentes. Puede seleccionar nuevos controles para añadir a la caja de herramientas, marcando la casilla que está a la izquierda del nombre del control que se quiere introducir. Una vez seleccionados todos los nuevos controles, hacer clic en APLICAR. Se sale de esta opción haciendo clic en ACEPTAR, y podrá observar que esos nuevos controles ya se le han añadido a la caja de herramientas.

Esos controles aparecen cada vez que cargue el proyecto actual. Para quitar controles de la caja de herramientas, se debe proceder de forma análoga, a lo que se hizo para ingresarlos en la caja de herramientas, pero al revés.

En el centro, y ocupando casi toda la pantalla, se encuentra el **Formulario**. Este Formulario es la interface gráfica de la aplicación, sobre el que se podrá añadir los controles que se necesite.

3.1.9 Ventana de Código.

La Figura 3.2 muestra un Formulario con su ventana de código. Cada objeto gráfico de VB tiene su propia ventana de código. Así, si en este formulario se hubiese introducido un Label y dos CommandButton, todos ellos tendrían su propia ventana de código. La ventana de código se puede ver haciendo doble clic sobre cualquier objeto del proyecto. En este caso se ha hecho doble clic sobre el único objeto que se tenía: el formulario.

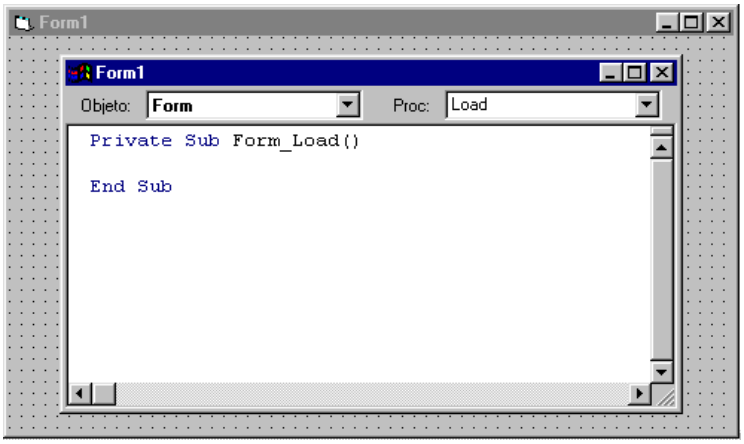


Figura 3. 2 Ventana de código.

Observando las dos cajas de la parte superior, uno con la inscripción **Objeto**: que para este caso tiene la palabra Form, y el otro con la inscripción **Proc:** (procedimiento), que tiene la palabra Load. A estas cajas se les denomina **Lista de Objetos** y **Lista de Procedimientos,** respectivamente.

Un **Evento** es algo que le puede ocurrir a un objeto. En una interface gráfica, lo que le puede ocurrir a un objeto es que se le haga clic, doble clic, que se pase el cursor del ratón por encima, etc. Éste es el **Evento**. El **Procedimiento** es la respuesta por parte de ese objeto, al evento que le está sucediendo.

Esa respuesta, es la forma de **Proceder** del objeto al evento que le está sucediendo, se debe programar según las necesidades, es decir, se debe escribir el código que necesite la aplicación como respuesta al evento que acaba de ocurrir. Posiblemente, no se quiere ninguna respuesta a muchos de los eventos que pueden acaecer a un objeto. Cada objeto tiene muchos eventos y solamente se quiere aprovechar los que interesan. Para que un evento no produzca ningún efecto, basta con dejar sin código el procedimiento correspondiente a ese evento. En los eventos que se desee que realice alguna operación, se escribirá en su procedimiento el código necesario para que esa operación se realice.

Para insertar un procedimiento se debe ir a la barra de menú, hacer clic sobre Herramientas, y en el menú que se desplegará, volver a hacer clic sobre Agregar Procedimiento. VB presentará un cuadro donde pedirá el nombre, si quiere que sea un

procedimiento, una función o una propiedad.

Un Módulo es una parte del programa donde solamente puede escribir código. Es igual que un formulario, sin interface gráfica. En un módulo pueden existir procedimientos al igual que en los formularios, pero como un módulo no tiene interface gráfica, esos procedimientos debe introducirlos el programador. El módulo tiene su propia ventana de código, al igual que un formulario, con un objeto único, el apartado "General". Aquí también se tiene la sección de declaraciones, al igual que los formularios.

Los módulos se emplean para la declaración de variables globales y para escribir el código de la aplicación que sea común a varios formularios. Para declarar variables que se pueden utilizar en todo el programa, se utiliza la sentencia de declaración **Public**. Esto evita tener que repetir código inútilmente. Ese código común se escribirá en un **procedimiento** que previamente se habrá insertado en este módulo, y se lo citará por su nombre desde cualquier parte del programa.

Para agregar un procedimiento se debe abrir una ventana de código (cualquiera) perteneciente al módulo o formulario donde se quiera insertar ese módulo, y a continuación hacer clic sobre **Herramientas | Agregar Procedimiento** de la Barra de Menú, como se representa en la Figura 3.3

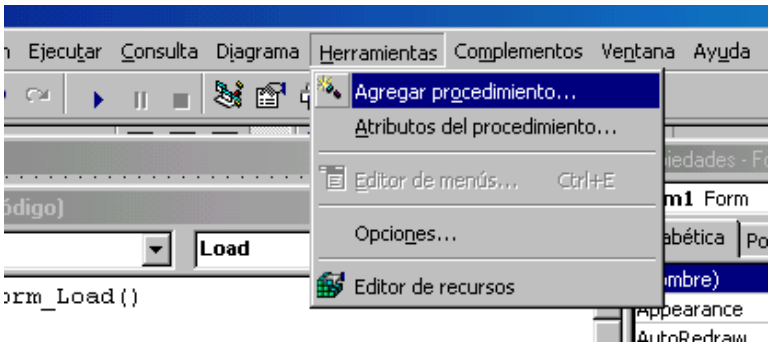


Figura 3. 3 Agregando un procedimiento.

Aparecerá un cuadro donde se debe poner el nombre de ese procedimiento y elegir el ámbito (Alcance en la figura) que se le quiere dar, público o privado, como indica en la Figura 3.4

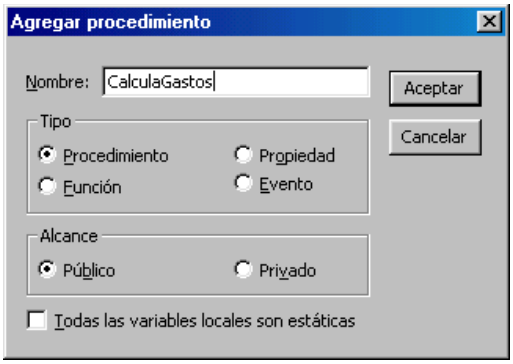


Figura 3. 4 Cuadro para agregar un procedimiento o función.

Si un procedimiento es público, se podrá acceder a él desde cualquier parte de la aplicación. Si es privado, solamente desde el módulo o formulario donde se ha introducido.

Para llamar un procedimiento se sigue las mismas normas que para las variables. Si se ha declarado como público en un módulo, se le citará por su nombre cualquiera que sea la parte del programa desde donde se lo cite.

3.1.10 Funciones.

Al igual que se introduce Procedimientos, se puede introducir Funciones en cualquier aplicación. Una Función es un Procedimiento al que se le envía uno o varios parámetros (o ninguno), y devuelve un valor. Al igual que los Procedimientos, pueden ser públicas (se verán en toda la aplicación) o privadas (se verán solamente en el formulario o módulo donde estén).

3.1.11 Main.

Si se tiene una aplicación que tiene 3 Formularios. En cada uno de ellos tiene código. Lógicamente la aplicación tendrá que presentar uno de ellos en primer lugar. Se deberá decirle a Visual Basic cual es el formulario inicial, y será ese por el que empiece. En ese formulario se dispondrá del código necesario para que la aplicación se ejecute en el orden deseado.

Si se hace clic en la Barra de Menú de Visual Basic, en **Proyecto | Propiedades** se obtendrá el siguiente cuadro de diálogo, que se indica en la Figura 3.5

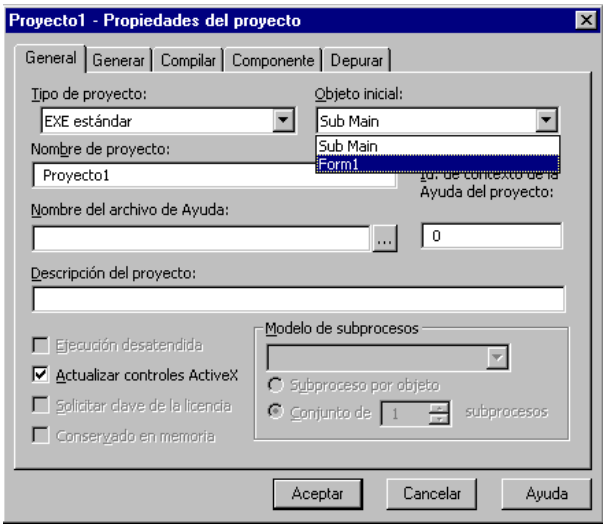


Figura 3. 5 Cuadro de propiedades del proyecto.

Se observa que tiene 5 pestañas, y actualmente tiene abierta la pestaña correspondiente a **General**, y tiene desplegada una lista donde se pide el Formulario Inicial. En esa lista figura también la expresión **Sub Main**. Si se selecciona como formulario inicial uno de los formularios, la aplicación comenzará por ese formulario. Si en uno de los **Módulos** existentes en el proyecto, se pone un procedimiento llamado **Main**, se puede comenzar la ejecución de la aplicación justamente por ese procedimiento. En ese procedimiento se pondrá el código necesario para que, posteriormente, se muestre uno de los formularios. Esto es muy práctico cuando se quiere hacer una función previa a mostrar cualquier formulario (abrir una base de datos, por ejemplo). Para comenzar la aplicación por **Main** se elige esa opción en la lista Formulario Inicial.

El cuadro de diálogo anterior sirve además para otros aspectos. Entre ellas, poner el nombre del proyecto (nombre que no aparecerá por ninguna parte, solo en los datos internos de la aplicación) y su descripción.

Existe otro cuadro similar en **Herramientas | Opciones** (Figura 3.6), donde se puede terminar de completar las condiciones de trabajo. En este cuadro se puede grabar el programa antes de ejecutar, lo cual es muy importante para evitar que, al ejecutarlo, por un código mal puesto se cuelgue el PC y se tenga que volver a empezar. En este cuadro se puede elegir también el formato de la ventana de trabajo, formato del editor, etc.

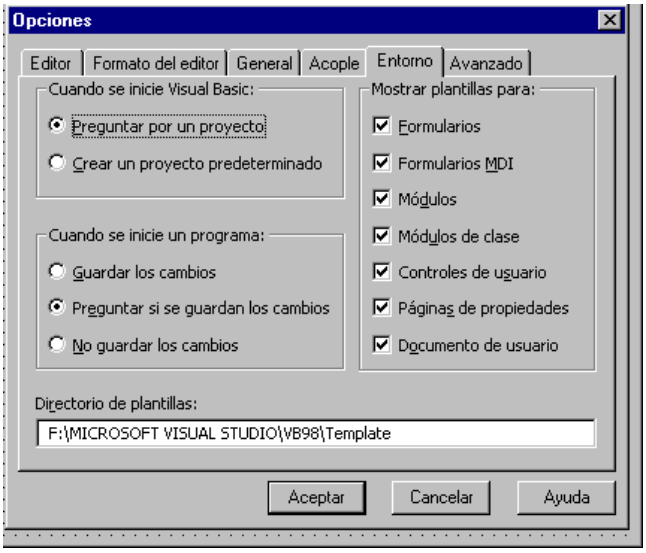


Figura 3. 6 Opciones del entorno de diseño.

En la pestaña Avanzado se puede cambiar el entorno de trabajo.

Se puede elegir también las ventanas que se quiere que estén siempre visibles, y que verifique automáticamente la sintaxis de una instrucción en el momento de escribirla.

La pestaña de Editor permite fundamentalmente 2 aspectos:
Observar solamente el código correspondiente al procedimiento que se está escribiendo ó ver el código del Módulo (o Formulario) completo.
En este último caso es un poco más difícil de manejar la ventana de código, sobre todo si el programador no está muy habituado a ello. Para seleccionar una u otra opción hay que seleccionar o deseleccionar la casilla Ver Módulo Completo.

3.1.12 El Formulario.

El primer objeto con el que se encontrará es el FORMULARIO. De hecho, cada vez que se inicie Visual Basic (VB) se presenta en pantalla un nuevo formulario, que tiene por defecto el nombre de Form1.
El **Formulario** es un objeto, que sirve de **soporte** de otros objetos. El nombre de FORMULARIO lo toma precisamente porque, al igual que un formulario de papel contiene textos escritos, lugares donde se puede escribir, figuras, cajas donde se puede elegir una u otra opción, etc.

En realidad un formulario es lo que normalmente se conoce por VENTANA. Cuando en vez de introducir datos, lo que se tiene es, por ejemplo, una ventana donde se reproducen exactamente los controles de operación de una máquina, parece en principio que sería más correcto el nombre de "ventana". En la Figura 3.7 se puede observar la pantalla de inicio de un formulario.

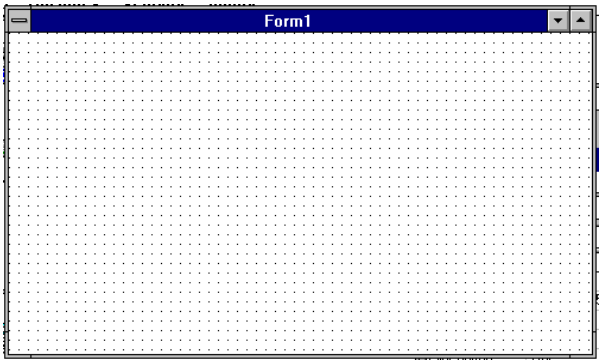


Figura 3. 7 Forma Inicial del Formulario.

Como cualquier objeto Visual Basic, un formulario tiene propiedades, procedimientos y métodos.

3.1.12.1 Propiedades del Formulario.

- **Name** (Nombre).- Define al objeto durante la ejecución del programa. Se introduce en tiempo de diseño y no se puede variar durante la ejecución. Nombre por defecto: Form1 (Form2 y sucesivos) Este nombre por defecto puede cambiarse, (debería cambiarse por norma en el momento de introducir el formulario), ya que de no hacerlo se corre el riesgo de borrar el código existente de otro formulario del mismo nombre en el momento de guardar la aplicación.

- **Caption** (Título).- Es el texto que aparecerá en la barra de Título cada vez que aparezca en pantalla este formulario. No tiene otra función dentro del programa. El programa no accede a este formulario por el título, sino por el nombre. Este título puede cambiarse en tiempo de ejecución.

- **Control Box:** Propiedad Booleana que admite los valores de True (verdadero) o False (Falso) (Valor por defecto: True). Si esta propiedad es True, aparecerá en la

esquina superior izquierda el icono para desplegar el menú de control de este formulario. Si esta propiedad se pone como False, no aparece dicho icono y por tanto no se puede desplegar dicho menú.

- **MinButton / MaxButton:** Botones de minimizar y maximizar este formulario. Son igualmente propiedades booleanas, que admiten el valor True o False (Valor por defecto: True**)**. Si están en True, aparecerá la flecha correspondiente. Si están en False, no aparecerá dicha flecha. Deben configurarse de una u otra forma, dependiendo si se quiere minimizar / maximizar este formulario durante la ejecución.

- **ControlBox**, **MinButton** y **MaxButton:** Para poder maximizar el formulario hijo. De no ponerlas a True, sí se pretende maximizar el formulario hijo (Propiedad WindowState=2), el formulario no aparece.

- **BorderStyle** (Tipo de borde).- Con el estilo del borde, lo que se puede conseguir es por ejemplo, que una aplicación no tenga ningún tipo de borde, que no se pueda cambiar su tamaño y que el tamaño se lo pueda variar.

  Dentro de **BorderStyle** se tiene 6 posibles opciones:

  0 - None: Permite que en una aplicación no aparezcan bordes.

  1 - Fixed Single: Se configura para que el borde de la aplicación siempre quede fijo. Con esta opción se puede poner los botones minimizar o maximizar según la aplicación.

  3 - Fixed Double: Con esta opción se puede incluir el menú de control, la barra de título, pero no se puede incluir ni los botones maximizar ni minimizar. Esta ventana no podrá cambiarse de tamaño.

  4 - Fixed Tool Window: Si se activa esta opción se mostrará un formulario con la fuente del título reducida. No se puede modificar el tamaño del formulario. Éste no aparecerá en la barra de tareas de Windows.

  5 - Sizable Tool Window: Se tiene una ventana de tamaño ajustable. El tamaño de la fuente del título aparecerá reducida. El formulario no aparecerá en la barra de tareas.

- **Backcolor** (Color del fondo).- Establece el color del fondo del formulario. Puede cambiarse en tiempo de ejecución.

- **Font** (Tipo de letra).- Especifica el tipo y tamaño de la letra que se usará en el formulario al utilizar el método Print. Valor por defecto**:** El determinado en la personalización. Al seleccionar esta propiedad en la ventana de propiedades, aparece un cuadro de dialogo donde se eligen ambos parámetros.

- **Icon** (Icono)**.-** Esta propiedad define el icono que va a representar a este formulario cuando esté minimizado. Valor por defecto: el icono que se le haya programado en la personalización. Si el formulario es el Padre o el de inicio de una aplicación, este icono es el que toma el Asistente de Instalación para colocarlo como icono de apertura del programa en el grupo de programas Windows correspondiente. Como valor de esta propiedad se le puede asignar directamente el icono o el nombre de un archivo (con su path correspondiente) que lo contiene, haciéndolo directamente sobre la caja de propiedades.

- **KeyPreview (**Propiedad Booleana).- Cuando un formulario tiene dentro de sí varios controles, uno de ellos es el que está activo. En estas condiciones, si se pulsa una tecla, esa pulsación la recibe en primer lugar el control que esté activo, y si éste no la procesa, pasa esa pulsación al formulario. Para hacer que esa pulsación pase previamente por formulario, debe ponerse esta propiedad en True. Esta propiedad se usará frecuentemente cuando quiera realizar alguna función pulsando una letra. Poner KeyPreview a True (Valor por defecto: False**)**, y puede conocer que tecla se ha pulsado en el procedimiento KeyPress del formulario. Si está en False, ese procedimiento no se ejecuta ya que la pulsación "no pasa" por el formulario.

- **Left** (Posición del Borde Izquierdo).- Indica la posición del borde izquierdo del formulario respecto a la parte izquierda de la pantalla. Normalmente no se introduce como valor numérico, sino que lo toma automáticamente de la posición que tenga el formulario en tiempo de diseño. Puede cambiarse en tiempo de ejecución, para mover el formulario.

- **MDIChild.-** Establece que este formulario es un formulario Hijo dentro de un formulario MDI (Valor por defecto: False). No se puede cambiar en tiempo de ejecución. Es una propiedad Booleana.

  True = Es formulario hijo False =No lo es

- **Picture** (Gráfico).- Mediante esta propiedad se puede ingresar un gráfico como fondo del formulario. El gráfico puede ser un bit-map o un fichero .ICO

- **ScaleHeight, ScaleWidth, ScaleMode. -** Indican la unidad de medida de dimensión de altura y anchura del Formulario. **ScaleMode** indica en qué unidades de medida se van a medir esas dimensiones. Acepta Twips (1), Point(2), Pixel (3), Character (4), Pulgadas (Inch) (5), Milímetros (6), Centímetros (7). Si se coloca la propiedad **ScaleMode** en cualquiera de estos valores, las propiedades **ScaleHeight** y **ScaleWidth** vendrán dadas automáticamente dependiendo del ancho del Formulario.

- **ScaleLeft, ScaleTop. -** Estas propiedades, medidas en la unidad de medida elegida para el ancho y alto mediante las propiedades **ScaleMode**, **ScaleWidth** y **ScaleHeight** anteriores, expresan las coordenadas iniciales de la parte izquierda y de la parte superior respectivamente del Formulario. Estas propiedades no afectan a la posición del Formulario en la pantalla (Si está maximizado seguirá ocupando toda la pantalla, si está en "Normal" ocupará el mismo sitio que se dio en tiempo de diseño).

- **Top (Posición del borde superior**). - Esta propiedad establece la posición del borde superior del formulario respecto a la parte superior de la pantalla (Objeto Screen). Normalmente no se introduce como valor numérico sino que lo toma automáticamente de la posición que tenga el Formulario durante el tiempo de diseño

- **WindowState.-** Establece el estado en el que aparecerá el formulario cuando se activa y presenta en pantalla. Admite tres opciones:

  - 0- Normal
  - 1- Minimizar
  - 2- Maximizar

3.1.13 Controles Elementales de Visual Basic.

Una vez introducido un Formulario, se pueden colocar los objetos (controles) que forman parte de la aplicación. Los controles, lo mismo que el Formulario, tienen **Propiedades, Procedimientos y Métodos.**

Para colocar un control en un Formulario, se debe seleccionar de la caja de herramientas existente en la parte izquierda de la pantalla de VB y llevarlo al Formulario. Si no existiese el control deseado en la caja de herramientas, se deberá obtenerlo de ***Proyecto*** | ***Componentes*** de la barra de menú de VB. Se elige el nuevo control, marcando la caja de opción a la izquierda de su nombre, y haciendo clic en ACEPTAR. Este control pasa a la caja de herramientas.

3.1.14 Command Button (Botón de Comando).

El Command Button es un objeto que sirve para introducir datos a través de la pantalla. En la Figura 3.8 el Botón de Comando tiene la siguiente forma:

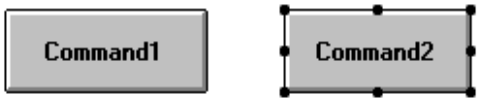


Figura 3. 8 Apariencia del Botón de Comando.

El botón de comando puede usarse para la entrada de datos, o para validar cualquier operación. El tamaño puede cambiarse a voluntad, pero la forma siempre es rectangular.
En la Figura 3.8 se observa dos botones de comando, uno de ellos (el Command2) marcado con unos puntos en su contorno, estos puntos permiten variar su tamaño en tiempo de diseño. También puede cambiarse su tamaño y posición en tiempo de ejecución.

Propiedades.

- **Name.-** Nombre
- **Caption (**Título). - Es el texto que aparece en el botón. Puede cambiarse en tiempo de ejecución. No tiene otra función dentro del programa.

- **Appearance.-** Apariencia
- **Backcolor (**Color de fondo). - Esta propiedad permite cambiar el color del botón cuando la propiedad Style está puesta a Graphycal. Si la propiedad Style esta puesta a Standars, esta propiedad solamente controla el color de un pequeño cuadrado que aparece rodeando el título.
- **Cancel . -** Establece un valor que indica si un botón de comando es el botón Cancelar de un formulario. Es una propiedad booleana, y admite los valores True o False. Puede utilizar la propiedad Cancel para dar al usuario la opción de cancelar los cambios que no se han hecho efectivos y devolver el formulario a su estado anterior. En un formulario sólo puede haber un botón de comando con la propiedad **Cancel** = True.
- **Causes Validation. -** Esta propiedad habilita o deshabilita la ejecución del evento **Validate** del control que tenía el foco antes de cambiar el mismo. El evento **Validate** de un control se ejecuta inmediatamente antes de que pierda el foco. *(Se dice que un control tiene el foco cuando es ese control el que está activado)*. Al hacer click sobre el botón de comando, siempre habrá algún control que pierda el foco para que lo tome el botón de comando. Si el botón de comando tiene la propiedad Causes Validation a True, se ejecutará el evento **Validate** de ese control que acaba de perder el foco. Si está a false, no se ejecutará.
- **Default. -** Establece un valor que determina el control CommandButton que es el botón de comando predeterminado de un formulario. Sólo un botón de comando de un formulario puede ser el botón de comando predeterminado. Cuando Default se define a True para un botón de comando, se define automáticamente a False para el resto de los botones de comando del formulario. Cuando la propiedad Default del botón de comando está establecida a True y su formulario primario está activo, el usuario puede elegir el botón de comando (invocando su evento Clic) presionando ENTRAR. Cualquier otro control que tuviera el enfoque, no recibe evento de teclado (KeyDown, KeyPress o KeyUp) de la tecla ENTRAR, a menos que el usuario haya movido el enfoque a otro botón de comando del mismo formulario. En este caso, al presionar ENTRAR se elige el botón de comando que tiene el enfoque en lugar del botón de comando predeterminado.

- **MaskColor.-** Establece cuál es el color de la imagen del botón que actuará como máscara para producir zonas transparentes. Para ello, la propiedad UseMaskColor debe estar a True.
- **Style.-** Define el estilo del botón, aceptando dos valores: Graphycal y Standard. Poniendo esta propiedad a Graphycal se puede cambiar el color del botón con la propiedad Backcolor e incluso introducir un gráfico (en la propiedad Picture, Downpicture o Disabledpicture).
- **UseMaskColor**.- Determina si el color asignado en la propiedad **MaskColor** se usa como máscara para crear zonas transparentes.
- **Eventos del botón comando.**

Clic	DragDrop	DragOver	GotFocus
KeyDown	KeyPress	KeyUp	LostFocus
MouseDown	MouseMove	MouseUp	

3.1.15 Label (Etiqueta).

Una etiqueta es un control que permite presentar un texto. La etiqueta debe usarse en aquellos casos en los que exista una información estática o dinámica que no deba ser cambiada por el operador.

En la Figura 3.9 la etiqueta puede adoptar estas formas: con borde tridimensional, borde plano o sin borde, y el texto justificado a la izquierda, a la derecha o centrado.

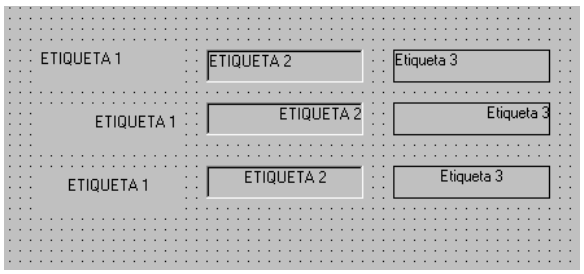


Figura 3. 9 Diferentes formas de borde para la etiqueta.

Se ha incluido la trama para poder observar los límites de la etiqueta sin borde.

Propiedades.

- **Alignment (**Justificación). - Establece si el texto se justifica a la izquierda, a la derecha, o se centra sobre la etiqueta.
- **Backcolor.-** Color de fondo
- **BackStyle (**Tipo de fondo). - Cuando se selecciona transparente, se ve solamente el texto de la etiqueta, dejando ver como fondo entre las letras, el color o Picture del formulario. Cuando se selecciona opaco, este texto se observa sobre un fondo gris.
- **BorderStyle (**Tipo de borde). - Sin borde o con borde. En caso de haber elegido en la propiedad
- **Appearance.-** El modo tridimensional, y eligiendo con borde en esta propiedad, el aspecto adopta una forma como incrustada en el formulario.
- **Caption.-** Título
- **DataSource, DataField. -** Establecen el **control data** asociado y el campo de la base de datos que se presentará en la propiedad **Caption**. Estas propiedades permiten presentar los datos de una Base de Datos mediante un procedimiento muy sencillo, con la ayuda de un control **Data**.
- **Eventos del Label.**

Clic	Change	DbClic	DragDrop
DragOver	LinkClose	LinkError	
LinkNotify			

3.1.16 Text Box (Caja de Texto).

Las cajas de texto son los controles con los que Visual Basic presenta o introduce textos. Es por tanto un control bidireccional. Normalmente se usan para introducción de textos, o para la presentación de aquellos que el operador pueda cambiar. Para cambiar o escribir un texto en una caja de texto, se debe conseguir que esa caja de texto tenga el foco y teclear el texto. Esto se puede lograr, haciendo clic con el ratón en esa caja de texto, o con la tecla TAB.

Las cajas de texto pueden tener una o varias líneas, según esté la propiedad **Multiline**. La

capacidad máxima de una caja de textos es de 64 Kbytes. La forma de una caja de texto es la siguiente (Figura 3.10), dependiendo de las propiedades BorderStyle y Appearance:

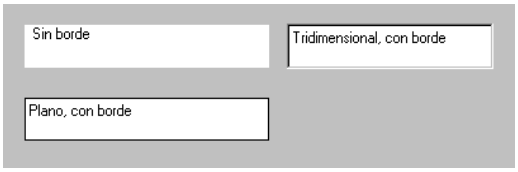


Figura 3. 10 Apariencias de la caja de texto.

Propiedades.

- **Alignment (**Justificación).- Establece si el texto se justifica a la izquierda, a la derecha, o se centra sobre la etiqueta. Esta propiedad solamente tiene efecto cuando la propiedad **Multiline** está a True, ignorándose en caso contrario. Es decir, permite colocar el texto justificado a la derecha cuando ese texto puede ser de varias líneas. Si la propiedad **Multiline** está a False, se justifica siempre a la izquierda.
- **BorderStyle (**Tipo de borde). - Sin borde o con borde. En caso de haber elegido en la propiedad.
- **Appearance.-** En modo tridimensional, y eligiendo con borde en esta propiedad, el aspecto adopta una forma como incrustada en el formulario.
- **DataField.-** Nombre de un Campo.
- **Enabled (**Habilitado). - Propiedad Booleana que habilita o deshabilita la etiqueta Cuando está deshabilitado (Enabled = False), no tienen efecto los eventos que se produzcan sobre el TextBox. No se puede escribir sobre él ni pasarle el foco, pero sí se le puede cambiar el texto mediante el programa. Puede ser una buena alternativa para impedir que el usuario pueda cambiar un determinado texto. Esta propiedad puede variarse en tiempo de ejecución.

- **Eventos de la Caja de Texto.**

Clic	Change	DblClic	DragDrop	DragOver
KeyDown	KeyPress	KeyUp	LinkClose	GotFocus

3.1.17 Frame (Recuadro).

Se obtiene directamente de la caja de herramientas. En la Figura 3.11 se puede observar en donde se encuentra el comando Frame.

Figura 3. 11 Posición del recuadro en la caja de herramientas.

En la Figura 3.12 se puede ver que tiene el siguiente frame:

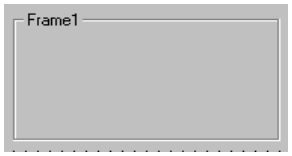


Figura 3. 12 Aspecto del recuadro en un formulario.

Un control Frame proporciona un agrupamiento identificable para controles. También puede utilizar un Frame para subdividir un formulario funcionalmente; por ejemplo, para separar grupos de controles **OptionButton**.

Cuando un control **Frame** tiene dentro otros controles, y se hace invisible al Frame, mediante su propiedad Visible = False, los controles interiores al Frame quedan también invisibles. Los controles Frame se utilizan para poner dentro de ellos diversos controles, que solo aparecerán en el programa cuando se cumpla una determinada condición. Cuando esa condición se cumpla, se pone la propiedad Visible del Frame a True y se verán todos los controles que tiene dentro. Si no se cumple esa condición, se pone la propiedad Visible del Frame a False y no se verá ni el Frame, ni los controles que contiene.

Propiedades.

- **ClipControls. -** Si esta propiedad está a **True**, los métodos gráficos en eventos Paint vuelven a dibujar el objeto entero. Antes del evento Paint se crea una zona de recorte

alrededor de los controles no gráficos del formulario. Si esta propiedad está a **False,** se dibujarán solamente las áreas últimamente expuestas. En tiempo de ejecución, esta propiedad es de sólo lectura.

- **Enabled. - C**uando esta propiedad está a **False**, tanto los procedimientos asociados al propio control Frame, como todos los controles dentro del Frame estarán inhabilitados. Si esta propiedad está a **True,** todos ellos están habilitados.
- **Visible. -** Cuando un Frame está con la propiedad **Visible** = **False,** tanto el propio Frame como todos los controles interiores a él serán invisibles.
- **Eventos del Frame.**

Clic	DblClic	DragDrop	DragOver
MouseDown	MouseMove	MouseUp	

3.1.18 Check Button y Option Button (Botones de elección y Opción).

Se obtienen directamente de la caja de herramientas, como se observa en la Figura 3.13

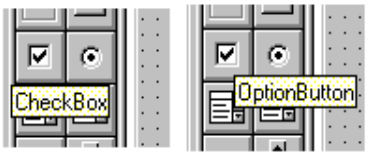


Figura 3. 13 Posición del Check y Option Buttons en la caja de herramientas.

Dada la similitud de ambos controles, se comentan conjuntamente.

El control **CheckBox**, o casilla de verificación, permite elegir una opción (activada/desactivada, True/False) que el usuario puede establecer o anular haciendo clic. Una X en una casilla de verificación indica que está seleccionada, activada, o con valor True. Cada casilla de verificación es independiente de las demás que puedan existir en el formulario, pudiendo tomar cada una de ellas el valor True o False, a voluntad del operador.

Un control **OptionButton** muestra una opción que se puede activar o desactivar, pero con

dependencia del estado de otros controles **OptionButton** que existan en el formulario. Generalmente, los controles **OptionButton** se utilizan en un grupo de opciones para mostrar opciones, de las cuales el usuario sólo puede seleccionar una. Los controles **OptionButton** se agrupan dibujándolos dentro de un contenedor como un control **Frame**, un control **PictureBox** o un formulario. Para agrupar controles **OptionButton** en un **Frame** o **PictureBox**, se debe colocar en primer lugar el **Frame** o **PictureBox** y, a continuación, arrastrar dentro los controles **OptionButton**.

Puede parecer que los controles **OptionButton** y **CheckBox** funcionan de forma similar, hay una diferencia importante: Cuando un usuario selecciona un **OptionButton**, los otros controles del mismo grupo **OptionButton** dejan de estar disponibles automáticamente. Por contraste, se puede seleccionar cualquier número de controles **CheckBox**.

Propiedades.

- **DataField / DataSource (**Propiedades del **CheckBox** solamente). - Establecen el control Data asociado y el campo donde están los datos (True / False) que se llevarán a la propiedad **Value**. Al igual que en los controles **Label** y **TextBox,** esta propiedad permite visualizar datos de una base de datos de una forma muy sencilla. En este caso, el **CheckBox** solamente permite presentar (logicamente) datos de tipo Booleano.
- **Height. -** Común a ambos controles, la altura del control solamente afecta a la colocación del título (puede ponerse un título largo en varias líneas), no al tamaño del botón, que es invariable.
- **Value (**Común a ambos controles). - El valor de esta propiedad indica si el control está seleccionado (Value = 1) o no seleccionado (Value = 0 ó False). Esta propiedad puede cambiarse en tiempo de ejecución. Mediante esta propiedad, se puede tanto forzar el valor, como leerlo.

  OptionButton se puede utilizar indistintamente las siguientes formas:

  Option1.Value = True Option1.Value = 1

  Con cualquiera de éstas, se seleccionará ese control por programa.

  Para el **CheckBox** solamente se podrá usar para este propósito la instrucción:

  Check1.value = 1

  Poner como Value el valor "True" dará en este control un error.

Para la lectura del Value, existe una diferencia entre el valor devuelto por el **CheckBox** y el devuelto por el **OptionButton**.

- **Atención.-** Presenta una diferencia entre uno y otro control, respecto a la forma de expresarse respecto a su valor cuando está seleccionado. Para forzar que el control NO esté seleccionado, o para leer el **Value** cuando no está seleccionado, se puede utilizar tanto **Value = 0** como **Value = False**. Sin embargo, cuando lo que se quiere es poner la propiedad a **True** hay una diferencia entre ambos controles.

  Para leer el dato existente en un **CheckBox** se utiliza la forma:
  variable = Check1.Value; donde variable tomará el valor 1.

  Para el control **OptionButton** se emplea la instrucción:
  variable = Option1.Value; donde *variable* tomará el valor True.

3.1.19 Timer (Temporizador).

Este objeto permite establecer temporizaciones. El control **Timer** solamente se ve durante el tiempo de diseño. En tiempo de ejecución, el control permanece invisible.
La temporización producida por el **Timer** es independiente de la velocidad de trabajo del ordenador (*Casi independiente, el timer no es un reloj exacto, pero se le parece*).

Se toma directamente de la caja de herramientas, y tiene el aspecto indicado en la Figura 3.14.

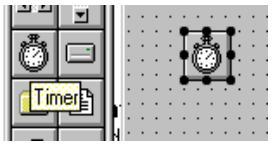


Figura 3. 14 Apariencia y posición del temporizador en un formulario.

Propiedades.

- **Interval. -** El valor de esta propiedad dará el intervalo de tiempo (en milisegundos) en que se producirá un evento **Timer** y consecuentemente, realizará el procedimiento asociado a este evento. Si el valor de la propiedad **Interval** está a 0 (Predeterminado),

no se produce el evento **Timer**. (El control **Timer** está **deshabilitado** cuando la propiedad **Interval = 0**)

3.1.20 Control Shape.

Se toma directamente de la caja de herramientas como se puede observar en la Figura 3.15:

Figura 3. 15 Posición del Shape en la caja de herramientas.

Shape es un control gráfico que se muestra como un rectángulo, un cuadrado, una elipse, un círculo, un rectángulo redondeado o un cuadrado redondeado.

Un control Shape no actúa como contenedor de controles. Es decir que un control **Shape** nunca servirá, por ejemplo, para albergar varios **OptionButton** y pretender que sean independientes de otros controles **OptionButton** que se encuentren fuera del control **Shape.**

Este control no tiene Procedimientos; solamente sirve para mostrar un determinado gráfico, ó envolver gráficamente a otros controles, pero no tiene ninguna aplicación en cuanto a programa.

Propiedades.

- **BackStyle. -** Esta propiedad establece si **Shape** permite ver a su través **(Transparent)** o no lo permite **(Opaque).**
- **BorderColor. -** Establece el color del borde.
- **BorderStyle. -** Establece el tipo de borde. Puede ser: Transparent (No se ve el borde), Solid (Borde de línea continua),Dash (línea a rayas), Dot (línea a puntos), Dash-Dot (línea de raya - punto), dash-Dot-Dot (línea de raya - punto - punto), InsideSolid (raya continua).

- **Shape. -** Establece su forma. Puede ser: Rectangular, cuadrado, redondo, ovalado, cuadrado con esquinas redondeadas y rectangular con esquinas redondeadas.

3.1.21 Control Line.

Se toma directamente de la caja de herramientas como se observa en la Figura 3.16.

Figura 3. 16 Posición del Control Line en la caja de herramientas.

Line, al igual que Shape, es un control gráfico que solamente sirve para poner una línea en un formulario. Del mismo modo, no tiene procedimientos, por lo que no sirve para aportar código al programa. Solo sirve para aportar una característica gráfica, es un adorno.

Propiedades.

- **BorderColor. -** Establece el color de la línea.
- **BorderStyle. -** Establece el tipo de línea. Puede ser: Transparent (No se ve la línea), Solid (Línea continua), Dash (línea a rayas), Dot (línea a puntos), Dash-Dot (línea de raya - punto), dash-Dot-Dot (línea de raya - punto - punto), InsideSolid (raya continua).

3.1.22 Common Dialog.

CommonDialog es un control del que se libran muy pocas aplicaciones. Normalmente se encuentra en la caja de herramientas. En la Figura 3.17 se presenta esta opción.

Figura 3. 17 Posición del Common Dialog en la caja de herramientas.

Este control no se presenta en tiempo de diseño más que con un simple icono, como se

indica en la Figura 3.18.

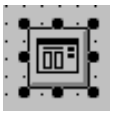


Figura 3. 18 Apariencia del Common Dialog en el formulario.

El cuadro de diálogo, **CommonDialog** se utiliza para varias funciones:

- Abrir Ficheros
- Guardar Ficheros
- Elegir colores
- Seleccionar Impresora
- Seleccionar Fuentes

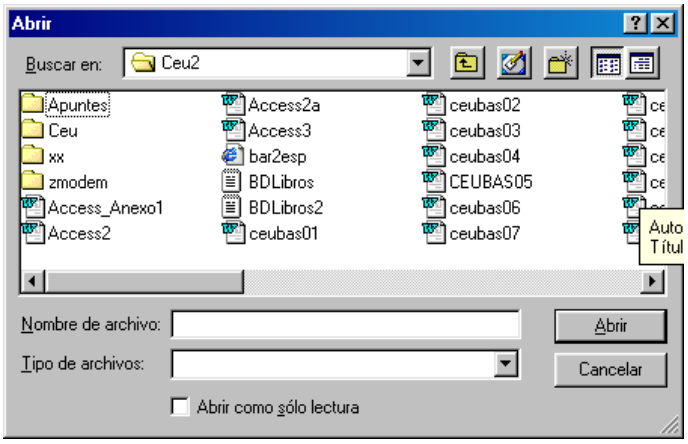


Figura 3. 19 Ventana de la Función Abrir en el Common Dialog.

En la Figura 3.19 se indica el CommonDialog mostrando la función Abrir. Note el icono para crear nueva carpeta, respecto del cual se puede indicar lo siguiente:

El cuadro de diálogo permite conocer datos con los cuales, y mediante el código adecuado, se puede abrir o guardar ficheros, elegir colores o seleccionar fuentes. Es decir, el CommonDialog **NO** realiza más funciones que mostrar ficheros existentes, fuentes

disponibles, colores.
El CommonDialog no es un control propio de Visual Basic, sino de Windows. Al introducir el control en la aplicación está introduciendo la versión más moderna del mismo.

3.1.23 Common Dialog para Abrir y Guardar Archivos.

Posiblemente las opciones más usadas del cuadro de diálogo son para abrir o guardar archivos. Para presentar el cuadro de diálogo correspondiente a Abrir Archivo, se debe ejecutar la instrucción:

Form1.**ShowOpen**

Para abrir el cuadro de diálogo para guardar un fichero, se debe ejecutar la instrucción:

FORM1.**ShowSave**

Pero antes se deberá introducir el filtro o filtros de archivos que necesite.
Un **filtro** sirve para que en el cuadro de diálogo se presenten solamente los ficheros de interés. Se pueden seleccionar por el nombre, la extensión o las dos cosas. La sintaxis para introducir un filtro es la siguiente:

FORM1.**Filter** = "Descripción del fichero|nombre.extensión"

Donde "nombre" puede ser un nombre o usar caracteres comodín. Lo mismo para "extensión"

Por ejemplo, para buscar archivos ejecutables, con cualquier nombre y extensión .EXE

FORM1.**Filter** = "Ficheros ejecutables |*.exe"

Se puede buscar varios tipos de ficheros, separando los caracteres de busca por medio del símbolo punto y coma.

FORM1.**Filter** = "Ficheros ejecutables |*.EXE;*.COM;*.BAT"
Puede también introducir dos filtros, separados por el separador "|" (En una línea

solamente), como se indica a continuación.

FORM1.**Filter**=”Ficheros de un tipo |*.EXE |Ficheros de otro |*.COM |Otros ficheros |*.BAT”

Por ejemplo: FORM1.**Filter** = "Ficheros Word |*.DOC|Ficheros Write |*.WRI"

En este caso, cuando se presente el cuadro de diálogo, solamente presentará uno de los filtros, y se deberá seleccionar manualmente el otro en un cuadro situado en la parte inferior izquierda del cuadro de diálogo. Para predeterminar cuál de los dos filtros saldrá sin tener que seleccionarlo, se ejecutará la siguiente instrucción, antes de invocar el cuadro de diálogo

FORM1.**FilterIndex** = 2

En este caso, aparecerá por defecto el filtro escrito en segundo lugar. El orden de los filtros comienza en 1.
El cuadro de diálogo deberá presentar un directorio. Este directorio con el que, por defecto, aparece el cuadro de diálogo, se puede introducir mediante la propiedad **InitDir**, que deberá contener el nombre del directorio y su path. Si no se especifica, comenzará por el directorio actual.

Para conocer el nombre del fichero elegido, se emplea la propiedad FileName :

Nombrefichero= FORM1.**filename**

El nombre del fichero, *Nombrefichero*, viene con su path, es decir, nombre y dirección completa, por lo que puede emplearse directamente la salida del **CommonDialog** para abrir un fichero.

El cuadro de diálogo de abrir o cerrar tiene la forma indicada en la Figura 3.20

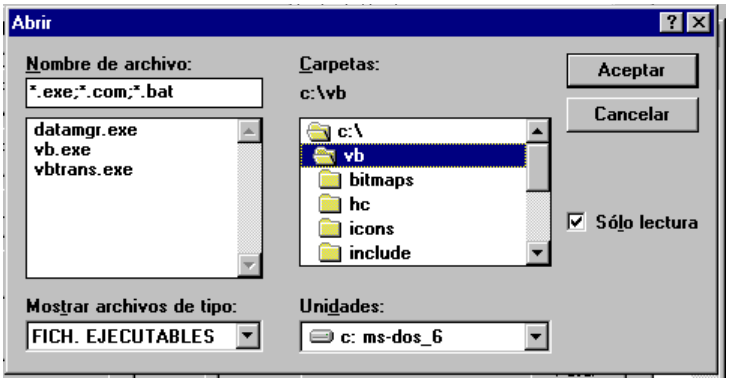


Figura 3. 20 Cuadro de Dialogo de Abrir o Cerrar.

El cuadro de dialogo "guardar" es similar. En este caso se ha desplegado la segunda opción de filtro de ficheros (Figura 3.21)

Figura 3. 21 Cuadro de dialogo para Guardar.

Como se observa ambos cuadros son prácticamente iguales. Se difieren en que uno pone **Abrir** y el otro **Guardar como** en su barra de título. También se diferencia en que el cuadro, **Guardar como** presenta los nombres de los ficheros con un tono apagado. Funcionalmente difieren; por ejemplo, si quiere guardar un fichero con un nombre ya existente, aparecerá la advertencia indicada en la Figura 3.22

Figura 3. 22 Cuadro de Dialogo para Archivo existentes.

3.1.24 Código del PLC en Visual Basic.

Como en todo autómata se necesita un interface hombre - máquina, en donde se puedan dar las instrucciones para que el PLC las ejecute; dentro de estas formas de comunicarse o de introducir instrucciones al autómata se conocen varias formas y las más conocidas son programación ladder, booleana, programación de alto nivel, y el diagrama de bloques, dentro de las cuales cada una tiene sus ventajas y desventajas, una respecto a la otra.

En la Tabla 3.1 se puede observar que el diagrama en escalera o ladder y el de bloques son muy buenos en cuanto a su aprendizaje, a la facilidad en el diseño del programa y al posterior entendimiento del mismo.

	LADDER	Booleano	Bloques	Alto nivel
Facilidad de aprendizaje del lenguaje de programación	Muy bueno	Muy bueno	Regular	Malo
Facilidad en el diseño del programa	Muy bueno	Malo	Bueno	Bueno
Elaboración de programas sencillos	Muy bueno	Muy bueno	Bueno	Bueno
Elaboración de programas complejos	Malo	Malo	Malo	Muy bueno
Elaboración de programas medianamente complejos	Regular	Regular	Bueno	Bueno
Comprensión posterior del programa	Muy bueno	Malo	Bueno	Malo
Tiempo de desarrollo de un programa	Mucho	Mediano	Mucho	Poco

Tabla 3. 1 Comparación entre algunos lenguajes de programación.

En función a la Tabla 3.1, la mayoría de PLCs constan de estas dos formas de lenguaje.

Pero ya en la práctica, debido a su capacidad de realizar programas un poco complejos, y de realizarlos en menor espacio, la programación en bloques saca ventaja; además en muchas industrias del sector eléctrico y petrolero, la mayoría de autómatas están

programadas en este lenguaje.

Como uno de los objetivos de este proyecto es realizar un PLC, en el cual se tenga las funciones básicas y que sea entendible al estudiante, y visto la necesidad de aprender otra forma de programación que sea entendible, tanto como el lenguaje ladder, se escogió este tipo de programación para el desarrollo del actual proyecto.

3.2 FUNCIONAMIENTO DEL PLC.

A nivel de software, un autómata se comporta de manera similar a una computadora; es decir, ejecuta una serie de rutinas de forma secuencial cada una con un objetivo determinado. Tanto en el PLC como en la computadora, el principal dispositivo de procesamiento es un microcontrolador, naturalmente con algunas pequeñas diferencias.

3.2.1 Lenguajes de Programación.

Los lenguajes de programación permiten al usuario generar rutinas o secuencias que una maquina pueda entender y ejecutar de manera automática. El software o conjunto de programas son la forma básica de comunicación, en el cual el usuario le indica a la máquina lo que desea que ella efectúe.

Existen muchos lenguajes de programación de los autómatas, entre los más principales están:

1. LADDER o diagrama en escalera
2. Mnemónico o de instrucciones booleanas
3. Diagramas de bloques de funciones
4. Instrucciones de alto nivel

Para entender cada una de éstas, se lo presentará con un pequeño ejemplo. Supóngase que se desea prender un foco con el siguiente circuito, en donde debe cumplir dos condiciones.

1. Que el interruptor S1 se accione
2. y el interruptor S2 no lo haga

Para mayor ilustración observe la Figura 3.23 del esquema eléctrico

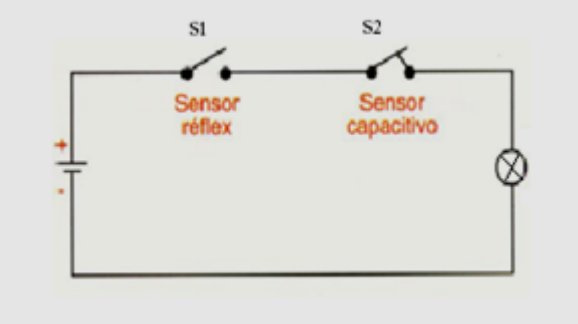


Figura 3. 23 Diagrama eléctrico para encender un foco.

A continuación se presenta la programación para los cuatro tipos de lenguajes (Figura 3.24)

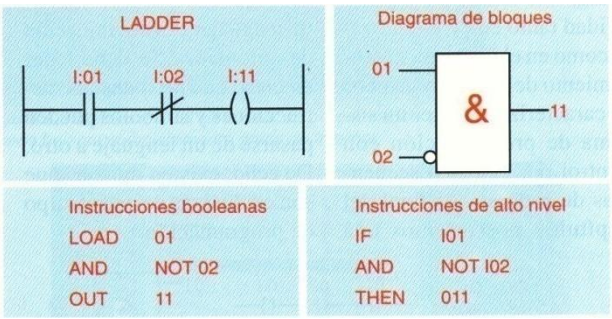


Figura 3. 24 El mismo proceso representado en 4 lenguajes de programación.

Como se observa en el lenguaje de diagramas de bloques, solo se utilizó una instrucción o un solo bloque para realizar el programa y es más fácil de entender con respecto a los otros lenguajes.

3.3 INSTALACIÓN Y MANEJO DEL SOFTWARE BYL.

El software BYL es un archivo auto ejecutable que sirve para la programación del autómata y además es un simulador en el cual se puede observar que salidas se activan de acuerdo al programa previamente establecido; el archivo es generado en Visual Basic el cual se puede instalar en cualquier computadora así ésta no tenga instalado Visual, por lo que solo se copia el archivo ejecutable en el disco duro de la computadora y dar doble clic en el icono respectivo.

Al abrir el software aparecerá la siguiente ventana de inicio, como se detalla en la Figura 3.25

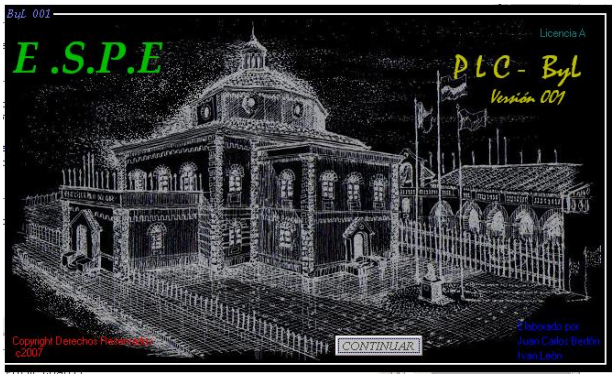


Figura 3. 25 Pantalla de inicio del programa ByL para el PLC.

Se presiona continuar y aparecerá la pantalla de inicio en donde se presenta la pantalla de programación, y la barra de herramientas y el menú, las cuales serán explicadas más adelante.

En la barra de herramientas están los iconos de nuevo, abrir, guardar, borrar, simular, detener y transferir al PLC; como se presenta en la Figura 3.26.

Figura 3. 26 Barra de Herramientas del Programa ByL.

3.3.1 Programación en Lenguaje de Diagrama de Bloques.

- **Entradas, Salidas.**

Los bloques hacen referencia específicamente a las entradas o condiciones que tiene un programa para que una salida se active. Cada bloque puede representar una entrada física del PLC, una memoria del programa, el estado de un temporizador o de un contador (Figura 3.27).

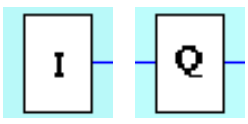


Figura 3. 27 Entradas y Salidas del Software ByL.

El icono hace referencia a las salidas o resultado de las operaciones indicadas en una línea, pueden representar la salida física del PLC, una posición de memoria, la activación o desactivación de un temporizador, el incremento de un contador.

Una particularidad de este tipo de programación es que los contactos o entradas pueden hacer referencia al estado de salida, sin tener que sacar otro bloque; a continuación en la Figura 3.28 un ejemplo en donde se activa una salida y éste a otra salida.

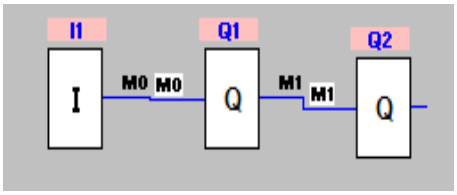


Figura 3. 28 Las salidas pueden usarse para activar otra salida.

- **Temporizadores.**

Otro componente de este software es el temporizador, que como su nombre lo indica, cada vez que alcanza un cierto valor de tiempo activa un contacto interno. Dicho valor de tiempo debe ser declarado por el usuario a través de la programación.

Existen dos tipos de temporizadores y diferentes formas de condición de arranque. En los temporizadores OFF DELAY se da una señal al relé del temporizador y el temporizador se activará cuando se haya cumplido el tiempo de preset. En el ON DELAY, el temporizador se activará inmediatamente y pasado el tiempo se desactivará, el cambio o conteo del tiempo solo se efectúa si se mantiene la alimentación al temporizador, de lo contrario el tiempo regresará a cero y no se activará el temporizador.

Para colocar el temporizador en el software, en la barra de componentes se da un clic en los temporizadores, y luego otro clic en donde se quiera insertar el temporizador; se debe señalar que solo se pueden insertar hasta un máximo de 10 temporizadores en cada programa.
En la Figura 3.29 se observa el ícono del temporizador mencionando anteriormente.

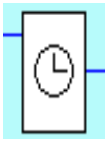


Figura 3. 29 Gráfico del temporizador en el software ByL.

Una vez insertado el temporizador, se le da un clic derecho; para setear sus parámetros, donde se puede escoger con qué tipo de temporizador quiere trabajar, ya sea ON u OFF DELAY; además del tiempo de preset, que puede ser desde 1 segundo hasta 24 horas.

En la Figura 3.30 se observa la caja de dialogo del temporizador para el seteo de los parámetros del temporizador.

Figura 3. 30 Seteo del temporizador en el software ByL.

A continuación en la Figura 3.31, se observa el uso de un temporizador, en donde con una entrada I1 se activa el T1 por 10 segundos, luego de este tiempo, T1 activa una salida Q1. Abajo se presenta el diagrama para el ejemplo indicado.

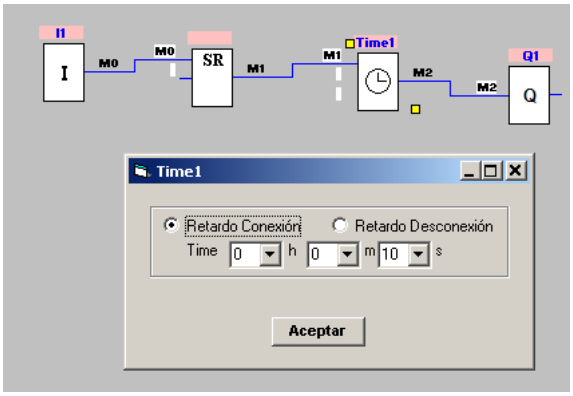


Figura 3. 31 Configuración de un temporizador.

- **Contadores.**

Al igual que los temporizadores los contadores tienen un valor predefinido llamado preset, el cual es el número que el usuario programa para que dicho contador se active. Por ejemplo, si un valor de preset es 5 y el valor de conteo es 4, se dice que el contador está inactivo sin que por ello no esté contando. Pero al dar otro pulso y el valor de conteo llegue a 5, el contador estará activo ya que llegó al valor preset.

Una vez que el contador se active, éste deberá ser reseteado; por lo que existe un reset por cada contador. Con el reset el contador se desactiva y vuelve a su estado normal y el conteo regresa a cero. Existen contadores ascendentes y descendentes.

Para colocar el contador en el software, en la barra de componentes se da un clic en los contadores y luego otro clic en lugar en donde se va insertar el contador. En la Figura 3.32 se presenta el símbolo del contador mencionado.

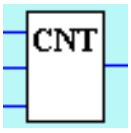


Figura 3. 32 Gráfico de los contadores.

Una vez insertado el contador, se da un clic derecho en el contador, para setear los parámetros, o sea el valor del preset del contador (Figura 3.33).

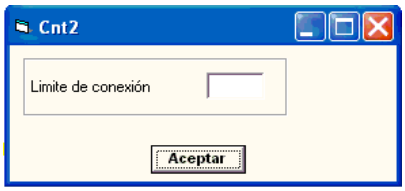


Figura 3. 33 Propiedades del contador.

A continuación se ilustra un ejemplo de contadores, en donde una entrada I1 activará al contador C1 luego de dar 4 pulsos, que fue el valor de preset. Cada vez que se pulse a I1 el contador incrementará su valor de conteo, al llegar al valor de preset el contador se activará y a la vez activará a una salida Q1, así permanecerá hasta que se dé un pulso en I2

que resetea al contador y se desactiva el contador y por ende Q1; o si se da un pulso en I3 el conteo descenderá hasta llegar a cero (Figura 3.34).

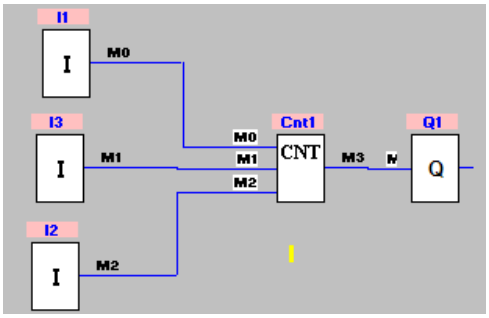


Figura 3. 34 Activación del contador y del reset.

- **Entradas y Salidas Análogas.**

Las entradas análogas (Figura 3.35) en un PLC son de gran utilidad ya que muchos dispositivos a los que se van a sensar son tipo análogo, los mismos que deben ser acondicionados para que se puedan procesar en el PLC.

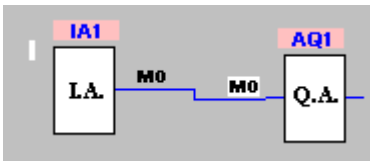


Figura 3. 35 Entradas y salidas análogas.

- **Comparador Análogo.**

El comparador análogo, como su nombre lo indica, compara dos señales análogas, y entregan a la salida una señal producto de la comparación (Figura 3.36).

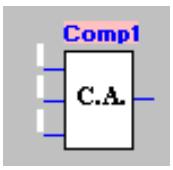


Figura 3. 36 Comparador analógico.

En el comparador analógico se puede fijar los valores mínimos y máximos de la magnitud física de las entradas; se puede seleccionar como se desea que comparare; es decir mayor, menor, mayor igual, menor igual o entre un valor determinado, y además seleccionar el estado de la salida, ya que puede enviar a apagar o encender una salida (Figura 3.37).

Figura 3. 37 Propiedades del comparador análogo.

En la Figura 3.38, se presenta un ejemplo del comparador. Utilizando las dos entradas análogas IA1 y IA2 hacia el comparador Comp1; éste comparará dichas señales con el operador mayor o igual a 50, y cuando suceda esto enviará a prender una salida Q1.

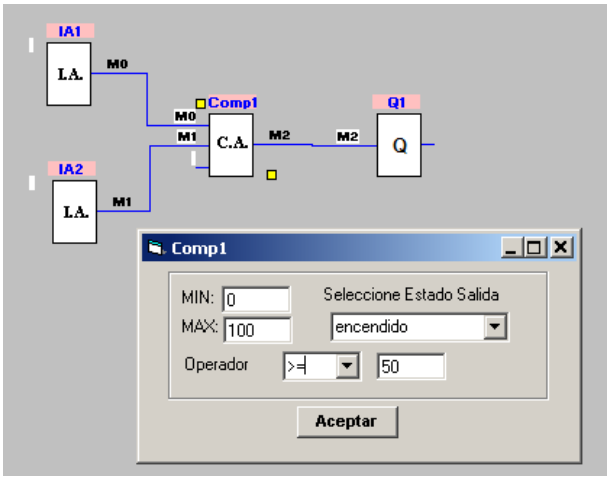


Figura 3. 38 Funcionamiento del comparador análogo.

- **Operadores Análogos.**

Al trabajar con señales análogas, a éstas se las puede comparar como ya se indicó anteriormente; pero también se las puede sumar, multiplicar o dividir como si se tratará de

valores numéricos, para lo cual la función de operador análogo ayuda a realizarlo (Figura 3.39).

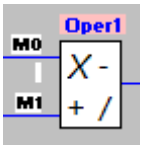


Figura 3. 39 Bloque operador análogo.

Dentro del bloque existe propiedades como el límite mínimo y máximo del rango, tipo de operación que se quiere realizar; esta por más decir que la salida es analógica.

En la Figura 3.40, se presenta un ejemplo del funcionamiento del operador análogo, en el cual se está sumando los valores de entrada; el resultado de esta operación va a la salida análoga.

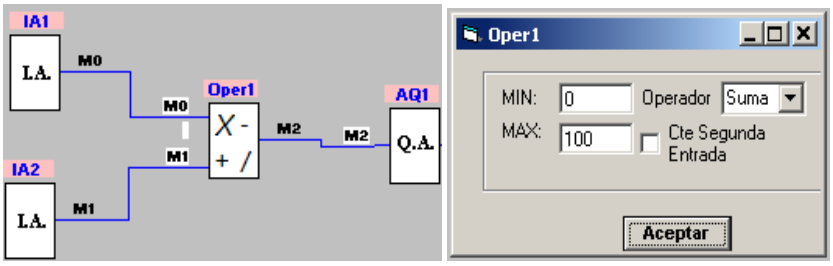


Figura 3. 40 Ejemplo del operador análogo.

- **RTC.**

El RTC o reloj calendario (Figura 3.41 y 3.42), es una herramienta que sirve en aplicaciones más reales, ya que éste es un reloj en tiempo real, al mismo que se le puede programar para que envíe una señal a una hora y día específicas.

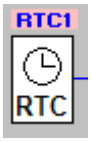


Figura 3. 41 Bloque de RTC.

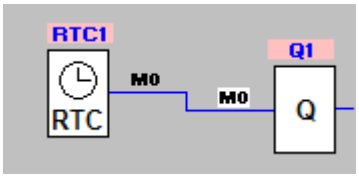


Figura 3. 42 Ejemplo de funcionamiento del RTC.

- **Simulación.**

A continuación se presentará la simulación del software; una vez que el usuario haya finalizado de programar, en la barra de herramientas se da un clic en **simular**, en donde aparece la ventana de simulación como se observa en la Figura 3.43

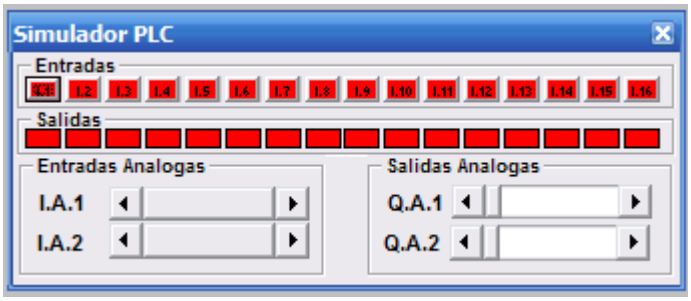


Figura 3. 43 Ventana de simulación del PLC.

Se puede apreciar que en la parte superior se encuentran las entradas y en la parte inferior se encuentran las salidas. Al momento de dar clic en una entrada que haya sido programada, empezará a simular lo programado por el usuario.

3.4 PROGRAMACIÓN DEL MICROCONTROLADOR BAJO PLATAFORMA VISUAL.

La programación del microcontrolador es prácticamente la programación para el envío de datos de Visual Basic hacia el microcontrolador PIC. (Programación en Visual Basic Anexo 2)

3.4.1 Comunicación con Visual Basic.

Para establecer comunicación serial con este software; es necesario agregar a la ventana de herramientas el icono de comunicaciones, Realizando un clic derecho en la ventana de herramientas, se selecciona **Componentes,** tal como se indica en la Figura 3.44

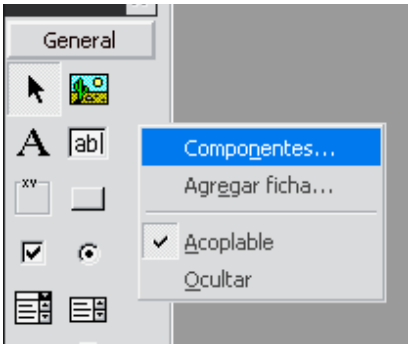


Figura 3. 44 Componentes de Visual Basic.

Al dar clic en **componentes** aparecerá una ventana en la que se encuentran varias pestañas (Figura 3.45), en la pestaña de controles se debe buscar el componente **Microsoft Comm Control 6.0** el mismo que se selecciona.

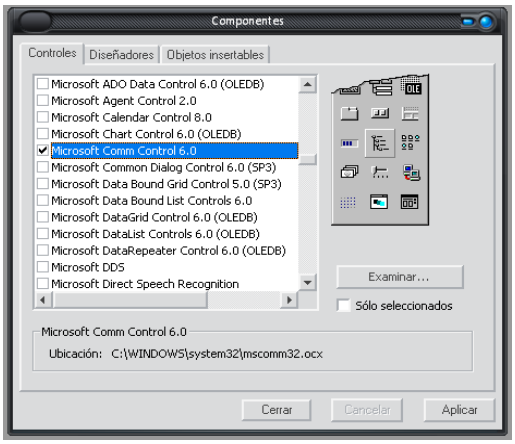


Figura 3. 45 Componente para la comunicación con el microcontrolador.

Al realizar este procedimiento se cargará en la caja de herramientas un nuevo elemento, el cual es similar a un teléfono analógico, y es el que permite la comunicación con el puerto serial que posee el computador. En la Figura 3.46 se puede observar el icono de comunicaciones.

Figura 3. 46 MScomm en la caja de herramientas.

Conocidos los elementos más importantes para el manejo de este software, se realizará un ejemplo el cual es similar a la programación que se realizó en el actual proyecto de Tesis.

- El primer paso será insertar los elementos que se utilizarán; en la Figura 3.47 se prenderá un led por cinco segundos y luego se apagará el led, por lo que se va a cargar en un nuevo formulario un shape (círculo) que simulará el led, un command button se encenderá el led, y un timer para dar intervalos.

Figura 3. 47 Componentes de VB para el ejercicio.

- Para setear la velocidad de envío y recepción de datos para la comunicación serial hay que ingresar a las propiedades del MScomm para la comunicación. Para esto se da un clic derecho en el componente y se selecciona propiedades y aparecerá la siguiente pantalla, como se indica en la Figura 3.48.

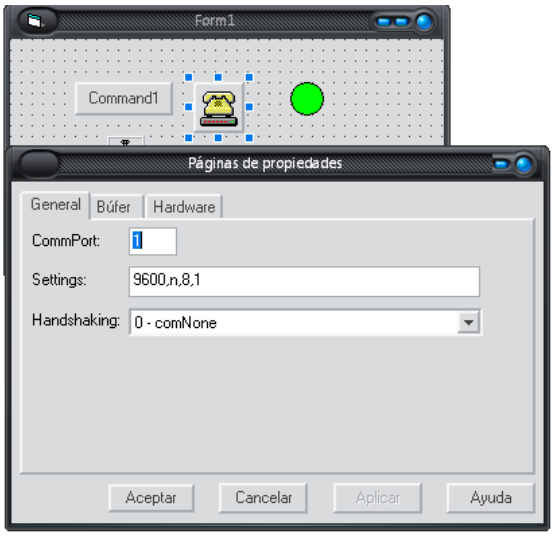


Figura 3. 48 Propiedades del MScomm para la comunicación.

- En la Figura 3.48 se observa el cuadro de propiedades del MScomm en el cual se indica los valores para la comunicación serial, los mismos que son: 9600,n,8,1; que significa que envía 8 bits de datos seriales, sin paridad y con un bit de parada, a una velocidad de 9600 bits por segundo; también permite seleccionar el puerto de comunicaciones que se va a utilizar en el ejercicio.

- A continuación se realiza el programa, como VB es una herramienta orientada a objetos, se empezará programando el formulario, para lo cual se dará un doble clic en el formulario y se escribe el siguiente código.

```
Private Sub Form_Load()
MSComm1.PortOpen = True
Timer1.Interval = 1
End Sub
```

El significado del anterior código es que al momento de ejecutarse el programa se habilite el puerto de comunicaciones y a la vez permita correr el timer a intervalos de 1 milisegundo.

- Luego se procede a programar el Command Button, el cual permitirá encender el led, simulando una entrada del PLC, por lo que se da doble clic en el comando y se escribe el código:

```
Private Sub Command1_Clic()
MSComm1.Output = "P"
End Sub
```

En esta parte se programa para enviar el dato serial “P” para que el microcontrolador (PIC) recepte este dato y encienda el led.

- Por último se programará el timer, en donde se podrá monitorear el led. Es decir, si el led se encuentra encendido el círculo cambiará a color rojo, y si está apagado el círculo estará de color verde. Al realizar doble clic en el timer, se escribe el siguiente código:

```
Private Sub Timer1_Timer()
DATO = MSComm1.Input  ' recepte el dato  y guárdelo en la variable DATO
If DATO = "E" Then           ' si la variable DATO = E entonces
Shape1.FillColor = vbRed    ' cambie el led a color rojo
End If                       ' indicando que el led se encendió

If DATO = "A" Then           ' si la variable DATO = A entonces
Shape1.FillColor = vbGreen   ' cambie el led a color verde
End If                       ' indicando que el led se apago
End Sub
```

3.4.2 Transferencia de datos del PC al PIC.

Éste es solo un pequeño ejemplo de cómo se pueden enviar datos desde VB hasta el PIC; en el proyecto actual de tesis, se están enviando un sinnúmero de cadena de caracteres por lo que la programación de envío se hace más compleja.

Por ejemplo, suponiendo que se desea activar una salida, con una entrada; luego enclavar la salida y finalmente apagar la misma. Para la programación del PLC en VB es muy fácil y ya es conocido, en la Figura 3.49 se observa como quedaría el enclavamiento.

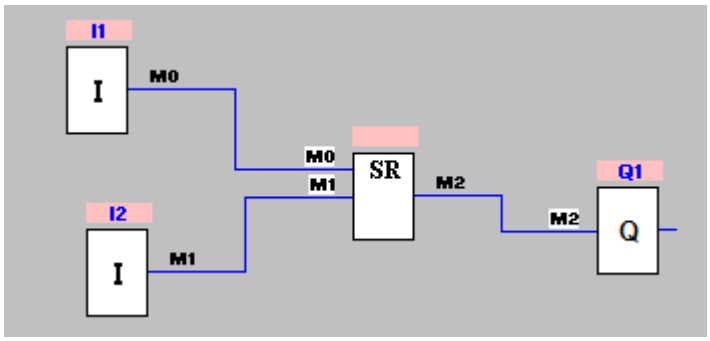


Figura 3. 49 Enclavamiento y reset de una salida con software ByL.

Se utiliza una entrada I1, un bloque de SET-RESET para el enclavamiento, una salida Q1 y por último una entrada I2 para apagar la salida; la programación en si en el software es sencilla, lo complejo es el momento de transferir estos datos al PIC.

Se va a utilizar el ejemplo anterior para entender en forma general cómo se transfieren los datos hacia el PIC.

Para la entrada I1, se tiene las variables: Que entrada se está usando (I1) y la variable de la línea de salida del bloque de entrada (0), por lo que se obtiene 2 variables: I1, 0.

Para la entrada I2 se tiene las variables: I2, 1

Para las salidas se tiene las variables: que salida se está usando (Q1), y la línea de entrada del bloque de salida (2), por lo que se obtiene. Q1, 2

Luego para el bloque set-reset se tiene las siguientes variables: la variable de la función (3), la línea de entrada para el set (0), la línea de entrada para el reset (1), y la línea de salida del bloque (2), entonces se obtiene: 3,0,1,0,0,2

Se puede observar que además de las tres variables se encuentran otras dos más. Las cuales van con valor cero, ya que éstas no influyen en el programa y solo sirven para rellenar la matriz de envío que se está utilizando.
Por último el dato a enviar al PIC desde VB para este ejemplo quedaría así:

I1,0 I2,1 Q1,2 F1,3,0,1,0,0,3

Se puede observar que se ha agregado F1, el cual quiere decir que es la primera función, en donde solo se pueden enviar hasta un máximo de 30 datos, y como cada bloque consta de 6 variables, quiere decir que con una función se puede programar hasta 5 bloques fácilmente.

En la Tabla 3.2, se detallan cada una de las funciones que tiene el software y el tipo de variables que se deberán enviar hacia el PIC.

BLOQUE FUNCIÓN	VARIABLE FUNCIÓN	ENTRADA 1	ENTRADA 2	ENTRADA 3	INDICE	SALIDA
AND	1	1….16 o 255	1…..16 o 255	1…..16 o 255	0	1……16
OR	2	1….16 o 255	1…..16 o 255	1…..16 o 255	0	1……16
S/R	3	1……16 SET	1……16 RST	X	X	1……16
XOR	4	1……16	X	1……16	X	1……16
CNT	5	1……16 UP	1….16 DOWN	1……16 RST	ÍNDICE CNT	1……16
NOT	6	X	1……16	X	X	1……16
TIMER	7	1……16 CONTEO	1 ON DELAY 2 OFF DELAY	INDICE TIMER	1……16 RESET	1……16
COMPARADOR ANALOGO	8	1……16	1……16	INDICE COMPARADOR	1 ENCENDIDO 2 APAGADO	1……16
PID	9	1 …2	1 …2	X		1 …2
RTC						
OPERADORES		1 …2 (ANALOGAS)	1 …2 (ANALOGAS	X	0 SUMA 1 RESTA 2 MULT 3 DIVI	1 …2

Tabla 3. 2 Variables para envío de datos.

A continuación se programará el microcontrolador, por lo que se realiza el diagrama esquemático de la conexión del PIC con el puerto serial. En la Figura 3.50 se puede observar la conexión del DB9 para la comunicación, donde el pin2 del DB9 es el receptor y el pin3 es el transmisor.

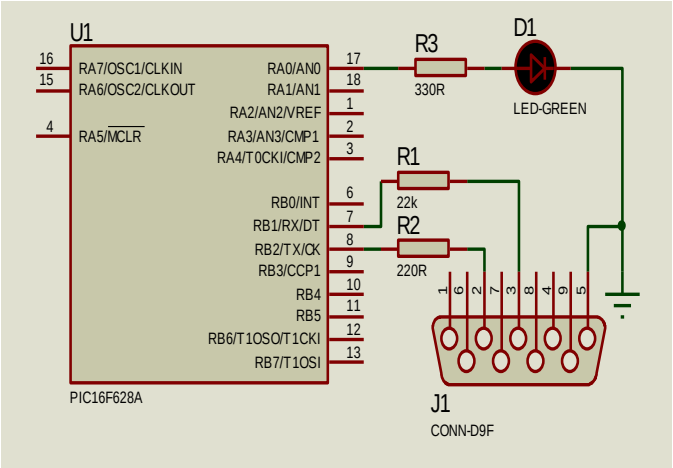


Figura 3. 50 Diagrama Esquemático de la comunicación serial DB9.

3.5 PROGRAMACIÓN DEL MICROCONTROLADOR (PIC) CON PIC BASIC PRO.

3.5.1 Microcode Studio.

PIC Basic Pro es un compilador el cual permite realizar la programación de un microcontrolador en un lenguaje de alto nivel, lenguaje Basic. A más de este compilador se dispone del software MicroCode Studio, el cual es un editor de texto exclusivo para facilitar la programación de los microcontroladores PIC. Por lo tanto MicroCode y PIC Basic Pro van juntos (Programación en MicroCode Anexo 3).

En la Figura 3.51 se observa la pantalla principal de MicroCode, donde se puede apreciar la barra principal y la barra de edición la cual se estudiará rápidamente.

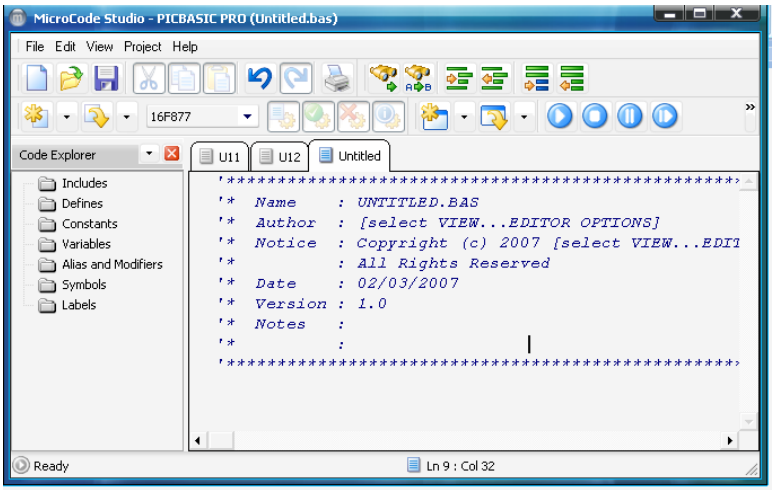


Figura 3. 51 Pantalla principal de MicroCode Studio.

3.5.2 Herramientas de MicroCode.

En la Figura 3.52 se encuentran los elementos, de abrir, cerrar, nuevo, pegar, cortar, copiar, undo; como estos elementos son bien conocidos no se analizarán uno por uno y solo se mostrará la Figura de cómo se encuentra en el software MicroCode.

Figura 3. 52 Barra principal de MicroCode Studio.

En lo que tiene que ver con la barra de edición y programación, en la Figura 3.53 se presentan los más importantes.

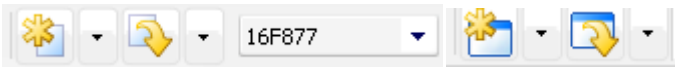
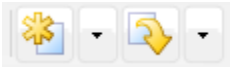


Figura 3. 53 Barra de edición de MicroCode Studio.

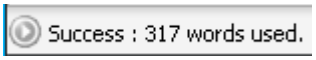
Esta ventana permite al usuario seleccionar el microcontrolador que va a utilizar para su programación, en este caso al momento de abrir el MicroCode se cargará con el PIC 16F877.



Estas dos herramientas son las más importantes al momento de realizar un proyecto, ya que permiten realizar la compilación del programa, se analizará detalladamente cada una de ellas.

Botón de solo compilar el programa, se lo puede también activar digitando F9, desde el teclado. Una vez compilado el programa el MicroCode genera 4 archivos: .ASM, .MAC, .PBP y .HEX, este último el más importante ya que este archivo permite grabar el microcontrolador PIC, así como también sirve para cargar en el software Proteus para poder simular el proyecto. Es importante decir que al momento de compilar también se graban los cambios hechos en la programación.

Con este botón se tiene dos funciones, ya que además de compilar el programa y generar los cuatro archivos que se mencionó anteriormente, también llama al programador IC-PROG, el cual permite grabar el PIC; se lo puede activar pulsando F10.



Luego de compilar, aparecerá en la parte inferior esta ventana que permite visualizar que espacio ocupó el programa en el PIC, como se puede observar se ha ocupado 317 palabras, de las 2048 palabras disponibles en el PIC 16F628A, (el PIC 16F877A dispone de 8192 palabras).

Explorador de Código.- Permite visualizar las variables, subrutinas, constantes, etc. Las cuales se generan en la programación, según se hayan utilizado, con la finalidad de encontrar o seleccionar rápidamente algún código que se necesite cambiar o editar como se observa en la Figura 3.54

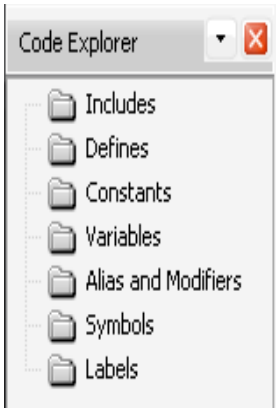


Figura 3. 54 Explorador de código.

Cuando se crea un nuevo proyecto, en la parte superior de la pantalla de edición, siempre aparecerá esta ventana de encabezado para cada uno de los programas que se realice, en donde se puede incluir el nombre del proyecto, nombre del autor, notas, etc.; como se observa en la Figura 3.55

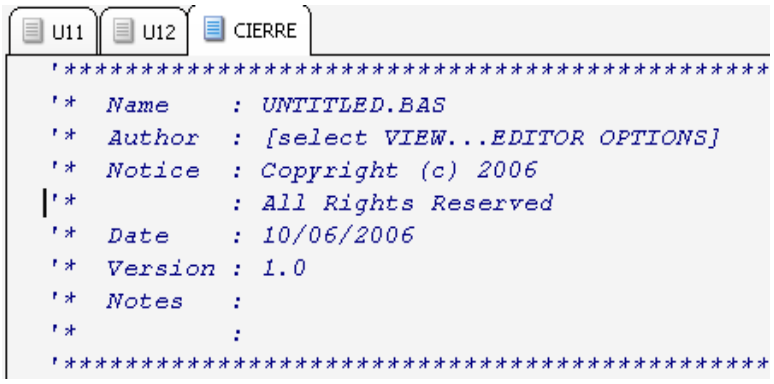


Figura 3. 55 Encabezado para cada proyecto.

3.5.3 Manejo del Software IC – PROG.

El IC-PROG es una herramienta fundamental para los microcontroladores en general, ya que éste permite grabar el archivo .HEX en el microcontrolador. Este programador permite seleccionar dispositivos como puerto serial, puerto paralelo, que permiten realizar la interfaz entre el computador y el microcontrolador. En la Figura 3.56 se aprecia la pantalla del IC - PROG en donde se observa que aparece 3FFF en todas las direcciones, lo que quiere decir que no se encuentra ningún archivo cargado.

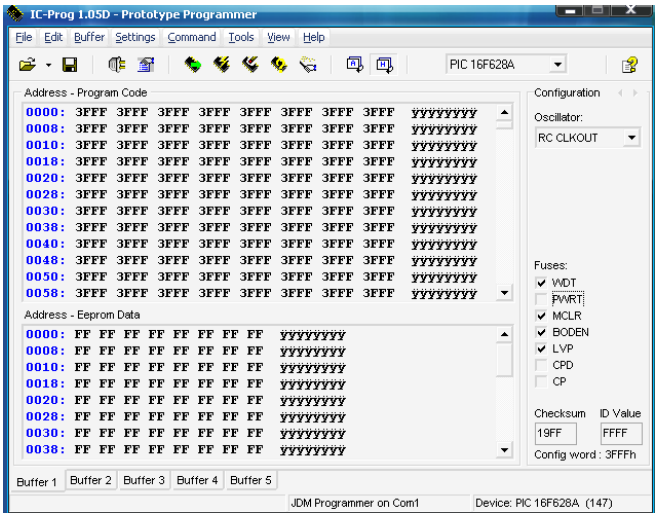


Figura 3. 56 Pantalla principal del IC-PROG.

Utilizando el ícono "abrir" del la figura anterior, se busca la carpeta en donde se grabó el programa y se selecciona el archivo .HEX que se desea cargar en el PIC, antes se debe escoger el PIC con el cual se va a trabajar; luego de haber cargado el programa, los códigos cambiarán (Figura 3.57).

Como se observa, al cargar el programa el archivo .HEX cambian los valores de dirección, por otros distintos a 3FFF.

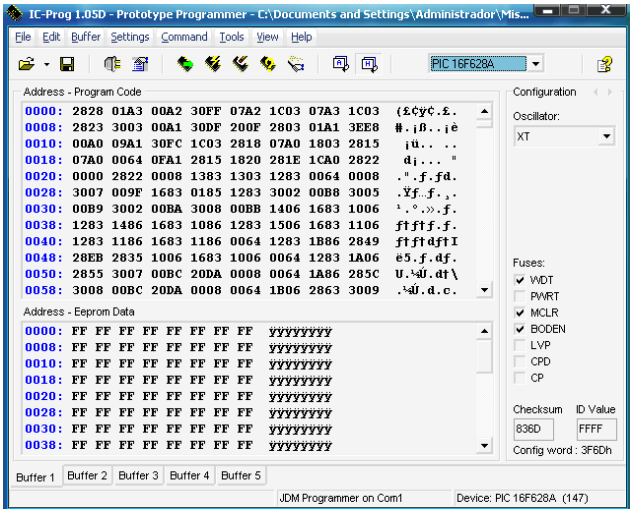


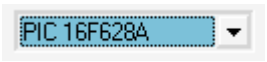
Figura 3. 57 Archivo .HEX cargado en IC-PROG.

En la barra de herramientas de la pantalla principal de IC-PROG, los iconos más importantes para el manejo del software, son los siguientes:

 Permite leer el contenido del Microcontrolador PIC

 Realiza la programación del microcontrolador, es decir carga el archivo .HEX al PIC

 Permite borrar el contenido del microcontrolador y dejarlo en blanco para poder ser grabado nuevamente



En esta ventana se escoge el microcontrolador que se va utilizar en el grabado del PIC.

En la ventana de configuración del oscilador de la pantalla principal de IC-PROG, se puede seleccionar la configuración deseada, es decir se puede cambiar el cristal interno de 4MHz que posee el PIC, por la que necesite el usuario (Figura 3.58).

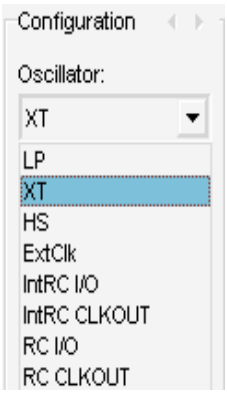


Figura 3. 58 Configuración del Oscilador.

En la ventana de configuración de Bits de la pantalla principal de IC-PROG, es recomendable desactivar la opción del MCLR, y dejarlo como se muestra en la Figura 3.59.

Figura 3. 59 Configuración de Bits.

Para que IC-PROG funcione de la manera correcta se deberá seguir los siguientes pasos:

- Seleccionar el microcontrolador PIC a utilizar.
- Cargar el archivo .HEX a IC-PROG.
- Configurar el oscilador a los requerimientos del usuario.
- Seleccionar los bits de configuración a los indicados anteriormente.

Un error puede suceder por dos motivos:

- No se encuentra bien habilitado el Hardware a utilizar.
- El microcontrolador que está utilizando esta en mal estado.

Es recomendable tener mucho cuidado en el manejo de los microcontroladores, ya que éstos son de tecnología CMOS; es decir, son muy vulnerables a la estática, por lo hay que manejarlos con pinzas para evitar el contacto con las manos.

3.6 PROGRAMACIÓN DEL MICROCONTROLADOR CON MIKROBASIC.

3.6.1 MikroBasic.

Microchip ha creado varias herramientas para la programación de microcontroladores ante las exigencias de los programadores, tales como Microcode, MicroC, Microladder, y ahora MikroBasic, la cual es una herramienta que en comparación a Microcode tiene mejores ventajas. MikroBasic, es una herramienta para realizar programación de microcontroladores, con un lenguaje parecido más al Basic como VB, de ahí su nombre. Para quien sabe programar en visual o en lenguaje C esta herramienta será muy útil y fácil de entender. En la Figura 3.60 se observa la pantalla principal de MikroBasic.

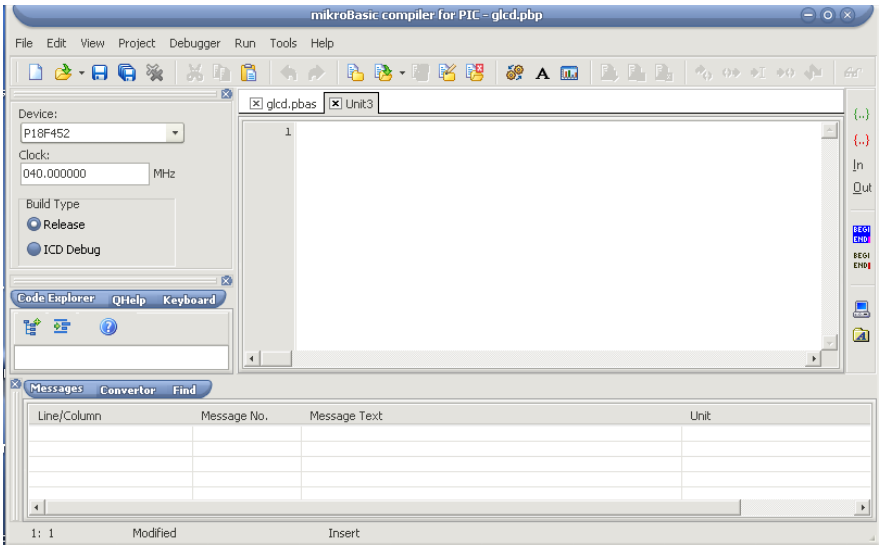


Figura 3. 60 Pantalla de Mikrobasic.

El software tiene su barra de herramientas y de menú, las cuales no son necesarias estudiarlas, por lo que solo se señalará, el uso para este proyecto. Como se mencionó anteriormente, éste tiene ventajas ante Microcode, y una de ellas es que se puede trabajar con LCD gráficos, como es el caso del LCD gráfico de 128x64 para la simulación de las variables de entrada y salida del PLC.

Primero se va a graficar el diseño que se quiere que aparezca en el LCD, por lo cual se utiliza el software Paint, ya que se grafica por puntos como lo realiza el LCD, en la Figura 3.61 se observa el diseño del PLC en Paint.

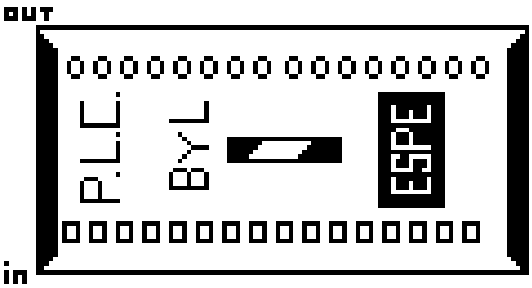


Figura 3. 61 Gráfico PLC que aparece en el LCD.

En MikroBasic, en la barra de herramientas en Tools/Editor GLCD, se selecciona el tipo de LCD, el que se escogerá es el LCD con configuración KSO108 y el tamaño 128x64 y se carga el gráfico (Load BMP) y luego se crea el código de MikroBasic (Create Code) como se observa en la Figura 3.62.

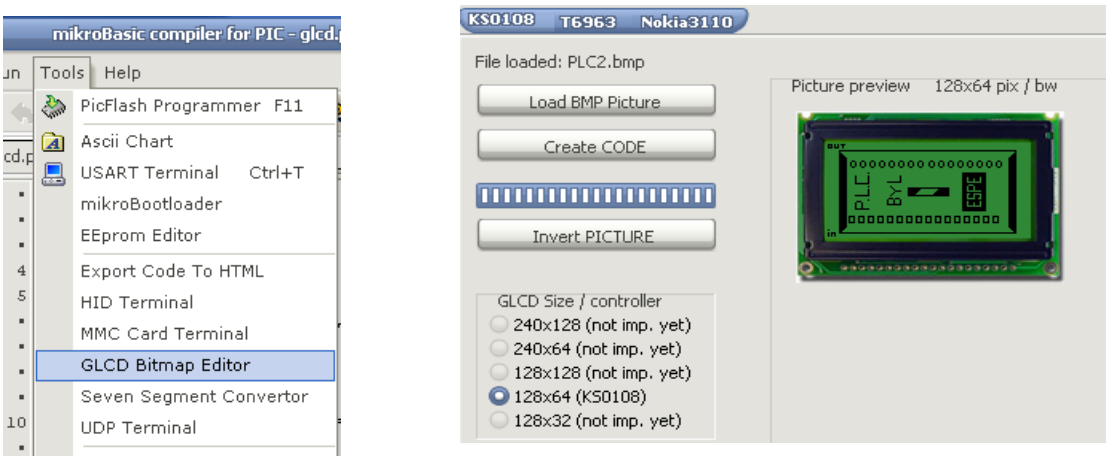


Figura 3. 62 Editor imágenes LCD.

Luego de haber creado el código se crea un módulo en MikroBasic y se guarda con un nombre, como en la Figura 3.63

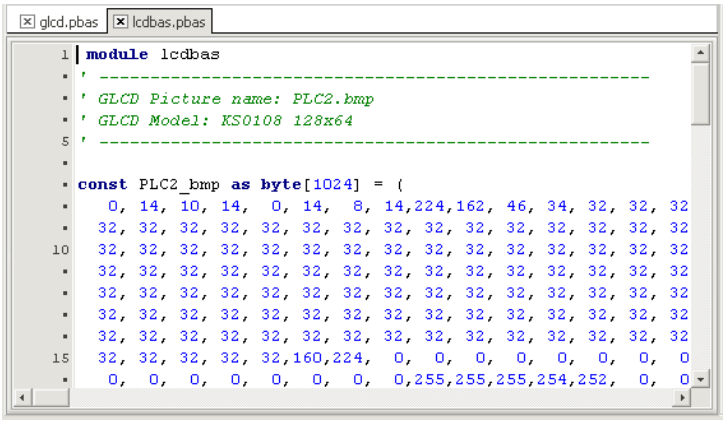


Figura 3. 63 Módulo del gráfico LCD.

Se observa que el gráfico se ha convertido en códigos, que son coordenadas, en donde van los puntos para crear el gráfico que se seleccionó, también se puede observar el nombre del gráfico y del LCD Gráfico que se está ocupando (Características GLCD Anexo 4).

Ahora se crea un nuevo proyecto, en la barra de menú, se da un clic en **nuevo proyecto** y aparecerá la siguiente ventana (Figura 3.64) en donde se editan las propiedades del proyecto como es el nombre, la dirección en donde se va a grabar, la velocidad del reloj, el PIC con el que se va a trabajar, así como la configuración de bits y del oscilador.

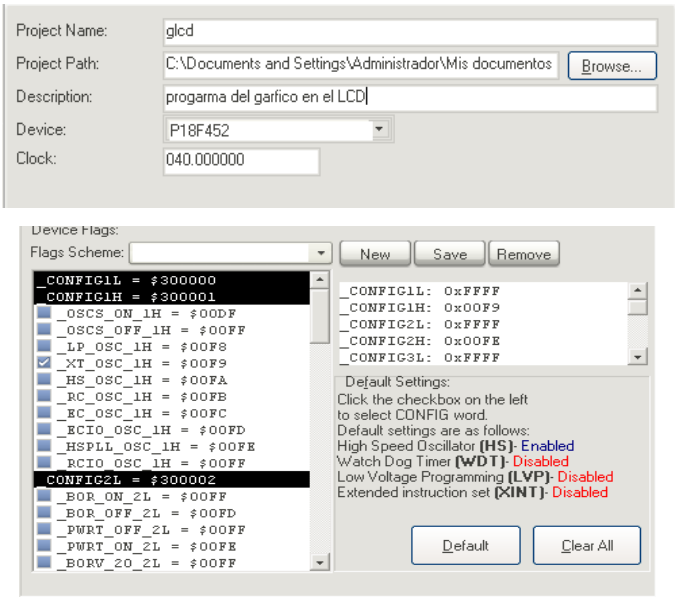


Figura 3. 64 Ventana de configuración de nuevo proyecto.

Al aceptar las configuraciones aparece una ventana en blanco para el nuevo proyecto y ahora se escribirá el código para el gráfico:

```
program glcd
include "lcdbas"
 main:
  GLCD_INIT(PORTB,2,0,3,5,7,1,PORTD)
GLCD_IMAGE(PLC2_bmp)
End.
```

En la primera línea aparece el nombre del proyecto, el que se digitó en la ventana de configuración; luego se incluye el **module** del gráfico que antes se creó, y luego la programación en sí; se inicia el LCD gráfico con los pines del PIC que se van ocupar; es decir, del puerto B se ocupa el pin 2, 0, 3, 5, 7,1 que son CS1, CS2, RS, RW, RST, E respectivamente y todo el puerto D.

Se procede a editar el circuito en Proteus y para su simulación. En Proteus en **herramientas**, en **LCD** existe el **Lcd ampire** que es el que tiene configuración KSO108 ya que éste es el que se ocupa en la programación del PIC, también debe notar que se está usando el PIC 18F452 que es el más útil para este tipo de proyectos con Lcd, también se puede utilizar otro PIC de la familia de los 18FXXX, como el que se ha utilizado para el proyecto que es el 18F4620.

En la Figura 3.65 se muestra la configuración del LCD y la conexión con el PIC en Proteus.

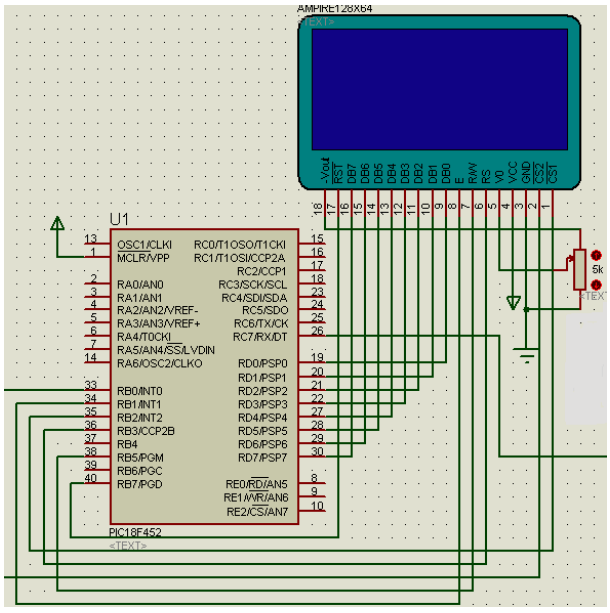


Figura 3. 65 Diagrama esquemático del LCD gráfico.

En la Figura 3.66 se indica la simulación en el LCD, de entradas y salidas del PLC.

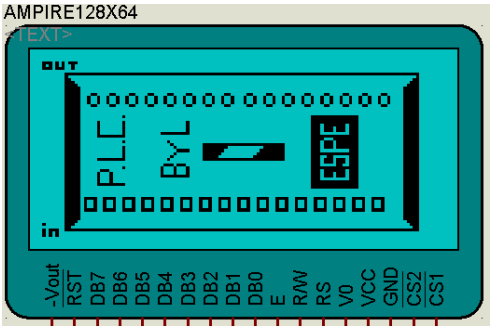


Figura 3. 66 Simulación PLC en LCD gráfico.

CAPÍTULO IV

4. IMPLEMENTACIÓN Y PRUEBAS DEL PROYECTO

4.1 GENERALIDADES.

En el presente capítulo se realizan las pruebas de implementación del proyecto una vez finalizado el software y hardware del PLC. Las pruebas a realizarse serán: Comunicación entre la PC y la tarjeta del PLC, pruebas de simulación del software y hardware, y la implementación física de la tarjeta a un proceso.

La comprobación del funcionamiento se la realizará en un prototipo en la cual se pueda observar el buen funcionamiento del PLC, para lo cual se escogió una maqueta de Bloquera, la misma que pertenece al Laboratorio de Oleoneumática, la cual se detallará más adelante.

4.2 PRUEBAS DE COMUNICACIÓN DEL PLC.

4.2.1 Comunicación USB - Serial.[27]

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora.

El puerto serial envía y recibe bytes de información un bit a la vez. Aún cuando es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias.

Por ejemplo, la especificación *IEEE 488* para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre dos dispositivos; por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros (Figura 4.1 Pines del DB9).

[27] **http://www.electronicaestudio.com/**

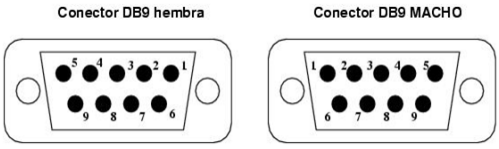


Figura 4. 1 Pines del conector DB9.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea, mientras se reciben datos por la otra. Existen otras líneas disponibles para realizar *handshaking*, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

Hoy en día existen situaciones donde es necesario convertir, o bien, emular un puerto serie RS232 a partir de un puerto USB como se muestra en la Figura 4.2. Esto se debe a que muchas de las computadoras modernas no incluyen el puerto serie, ya que para aplicaciones informáticas se considera obsoleto. Sin embargo existen muchas aplicaciones en electrónica donde resulta muy conveniente usar el protocolo RS232 para el intercambio de información y la PC resulta la interface más conveniente.

Figura 4. 2 Cable USB – RS232.

Lo que hacen estos adaptadores es emular un puerto serie mediante el puerto USB Estos adaptadores vienen con un software que una vez instalado crea un puerto serie virtual a través del puerto USB.

En la ventana de administrador de dispositivos de Windows, accesible desde el **panel de control/Sistema** se muestran los puertos COM y, previa instalación del controlador, se

observa el puerto Prolific USB-to Serial Comm Port, en este caso asignado como el COM10. El número de COM que se le asigna a este puerto se configura automáticamente en la instalación, pero se puede cambiar a cualquier otro que esté disponible, hasta 256; basta con ver las propiedades de ese puerto dando doble clic sobre el COM y posteriormente modificar el número (Figura 4.3). No puede haber dos puertos con el mismo número.

De esta manera se puede intercambiar información entre una PC y un dispositivo externo que utilice la norma RS232 mediante el puerto USB.

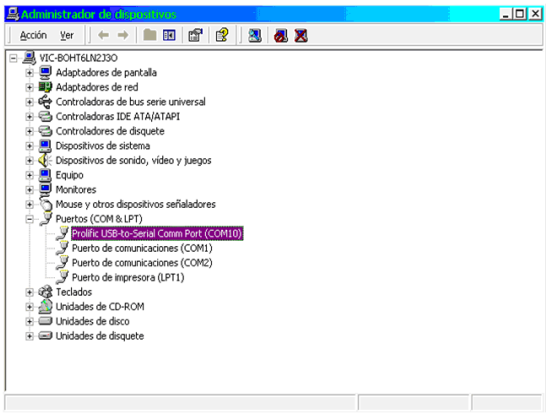


Figura 4. 3 Configuración del conversor USB-RS232.

4.2.2 El Puerto Serie.

En las comunicaciones seriales usando la norma RS232, se debe tomar en cuenta que la recepción de un dato ocurre de manera asíncrona; es decir, puede ocurrir en cualquier momento. El hardware de la computadora se encarga de interrumpir cualquier proceso que esté en ejecución cuando un nuevo dato llega al puerto. Es posible que el sistema operativo no pueda atender al nuevo dato de entrada, por lo que lo almacena en un buffer que contiene la secuencia de datos que van llegando.

Windows se encarga de la gestión de los puertos y se puede acceder a ellos de dos formas: Mediante las funciones de la API de Windows, o bien usando el control de comunicaciones de Visual Basic.

El método más recomendado es usar la API de Windows ya que fundamentalmente este sistema operativo es multitarea, y es conveniente que las aplicaciones no se apropien de los recursos del sistema, es recomendable dejar a Windows la tarea de la operación sobre el puerto.

Sin embargo Visual Basic permite manipular las comunicaciones seriales por medio de un control llamado Microsoft Comm Control. Mediante este control se gestionan las comunicaciones de manera simple y por medio de propiedades y métodos típicos de Visual Basic. Este control se agrega desde la ventana Components de Visual Basic (Figura 4.4)

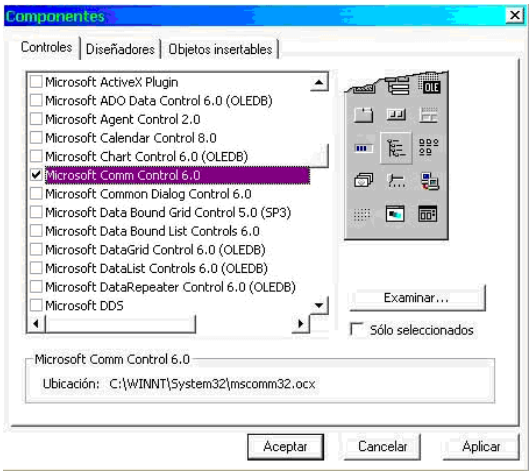


Figura 4. 4 Configuración Microsoft Comm control de Visual Basic.

Dentro de una aplicación, este control permite configurar las comunicaciones seriales mediante el puerto COM (Visual Basic 6 solo permite abrir hasta 16 puertos COM), el BaudRate, la paridad, el control de flujo, etc.

Una secuencia de cómo configurar un puerto en VB puede ser la siguiente:

Conexión.ComPort = 5 'Selecciona el puerto COM5

Conexión.Settings = "9600,n,8,1" ' BaudRate = 9600, n: sin paridad, 8 bits de 'informacion, 1 bit de parada.

Conexión.PortOpen = True; Abre el puerto (Figura 4.5 Configuración del puerto para Visual Basic).

Figura 4. 5 Configuración del puerto en Visual Basic.

Una vez entendido la configuración del puerto serial, se procederá a realizar la comunicación entre la PC y el PLC, como se muestra en la Figura 4.6

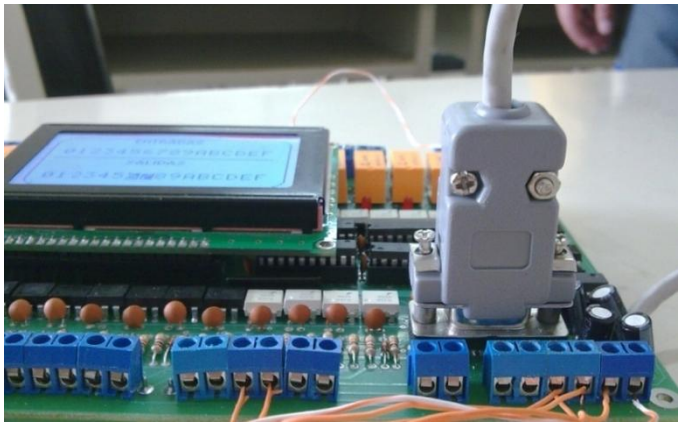


Figura 4. 6 Puerto serial del PLC.

4.3 PRUEBAS DE OPERACIÓN DEL PLC.

Para realizar las pruebas de operación del PLC, se implementará un proceso en el cual se pueda observar el funcionamiento de entradas, salidas y la aplicación del software del PLC.

Para la implementación, se utilizará un prototipo de Bloquera, como se observa en la Figura 4.7. A continuación se detalla su funcionamiento.

El proceso consta de electroválvulas, motores, pulsadores, relés, y se emplean en la parte neumática válvulas, pistones neumáticos, reguladores de caudal.

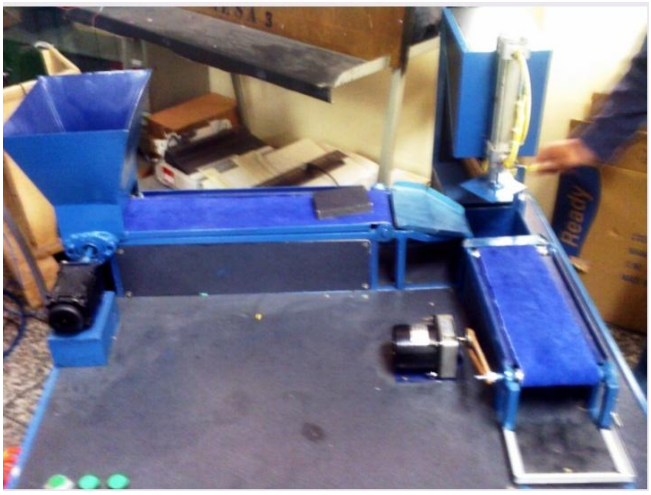


Figura 4. 7 Prototipo de una bloquera.

4.3.1 Descripción de los Componentes de la Bloquera.

4.3.1.1 Pistones de Simple y Doble Efecto.

- Un cilindro de accionamiento único usa aire comprimido para producir movimiento y fuerza. Tiene un pistón que puede deslizarse "hacia arriba" y "hacia abajo". Un muelle hace subir al pistón dentro del cilindro. Sin embargo, cuando la válvula se acciona, como se muestra en la Figura 4.8, el aire comprimido entra en el cilindro y le obliga a bajar su émbolo. El aire del otro lado sale por el orificio de escape.

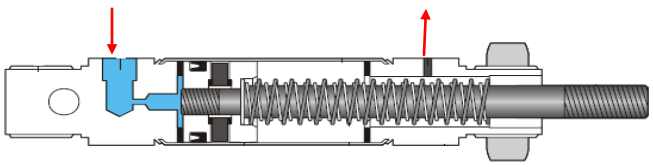


Figura 4. 8 Cilindro de simple efecto.

- El cilindro de doble efecto (Figura 4.9) significa que tanto el movimiento de salida como el de entrada, son debidos al aire comprimido; es decir, el aire comprimido ejerce su acción en las dos cámaras del cilindro, de esta forma realiza el trabajo en los dos sentidos.

El campo de aplicación de los cilindros de doble efecto es mucho más extenso que los de simple efecto, se lo construye siempre en forma de cilindro de émbolo y posee dos tomas para el aire comprimido situadas a ambos lados del émbolo.

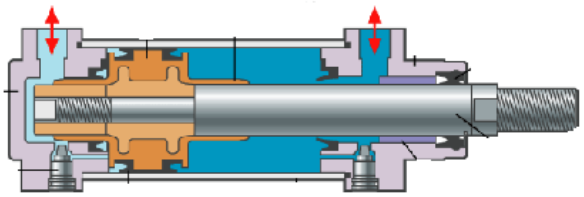


Figura 4. 9 Cilindro de doble efecto.

4.3.1.2 Válvulas Neumáticas.

Los mandos neumáticos están constituidos por elementos de señalización, elementos de mando y de potencia. Los elementos de señalización y mando modulan las fases de accionamiento de los elementos de potencia y se denominan válvulas. Los sistemas neumáticos e hidráulicos están constituidos por:

- Elementos de información.
- Órganos de mando.
- Elementos de trabajo.

Para el tratamiento de la información de mando es preciso emplear aparatos que controlen y dirijan el fluido de forma preestablecida, lo que obliga a disponer de una serie de elementos que efectúen las funciones deseadas relativas al control y dirección del flujo del aire comprimido.

En los principios de la automatización, los elementos rediseñados se accionan manual o mecánicamente. Cuando por necesidades de trabajo se precisaba efectuar el mando a distancia, se utilizan elementos de comando por símbolo neumático.

La gran evolución de la neumática y la hidráulica ha hecho, a su vez, evolucionar los procesos para el tratamiento y amplificación de señales, y por tanto, hoy en día se dispone de una gama muy extensa de válvulas y distribuidores que permiten elegir el sistema que mejor se adapte a las necesidades.

Hay veces que el comando se realiza manualmente, y otras obliga a recurrir a la electricidad (para automatizar) por razones diversas, sobre todo cuando las distancias son importantes y no existen circunstancias adversas.

Las válvulas en términos generales, tienen las siguientes misiones:

- Distribuir el fluido
- Regular caudal
- Regular presión

Las válvulas son elementos que mandan o regulan la puesta en marcha, el paro y la dirección, así como la presión o el caudal del fluido enviado por el compresor o almacenado en un depósito.

Según su función las válvulas se subdividen en 5 grupos:

1. Válvulas de vías o distribuidoras
2. Válvulas de bloqueo
3. Válvulas de presión
4. Válvulas de caudal
5. Válvulas de cierre

4.3.1.3 Válvulas distribuidoras[28]

Estas válvulas son los componentes que determinan el camino que ha de tomar la corriente de aire, a saber, principalmente puesta en marcha y paro (Start-Stop). Son válvulas de varias vías, los cuales determinan el camino que debe seguir el fluido bajo presión para efectuar operaciones tales como puesta en marcha, paro, dirección, etc.

Pueden ser de dos, tres, cuatro y cinco vías correspondiente a las zonas de trabajo y, a la aplicación de cada una de ellas, estará en función de las operaciones a realizar.

- **Representación esquemática de las válvulas.**

Para representar las válvulas distribuidoras en los esquemas de circuito se utilizan símbolos como se puede observar en la Tabla 4.1; éstos no dan ninguna orientación sobre el método

[28] **http://www.sapiensman.com/neumatica/neumatica14.htm**

constructivo de la válvula; solamente indican su función. Hay que distinguir, principalmente:

1. Las vías, número de orificios correspondientes a la parte de trabajo.
2. Las posiciones, las que puede adoptar el distribuidor para dirigir el flujo por una u otra vía.

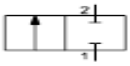
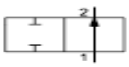
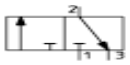
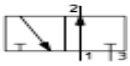
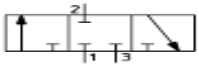
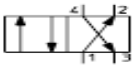
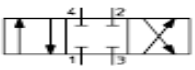
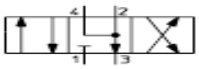
Símbolo	Descripción
	Válvula de control direccional Válvula de dos vías, cerrada posición normal
	Válvula de control direccional Válvula de dos vías, abierta posición normal
	Válvula de control direccional Válvula de 3 vías, cerrada posición normal
	Válvula de control direccional Válvula de 3 vías, abierta posición normal
	Válvula de control direccional Válvula de 3 vías, cerrada posición neutra
	Válvula de control direccional Válvula de 4 vías,
	Válvula de control direccional Válvula de 4 vías, cerrada posición neutra
	Válvula de control direccional Válvula de 4 vías, escape posición neutra
	Válvula de control direccional Válvula de 5 vías,

Tabla 4. 1 Simbología de las válvulas distribuidoras.

4.3.1.4 Válvula Reguladora de Caudal.

Muchas aplicaciones requieren que se utilice una velocidad menor para evitar un desplazamiento brusco, esto se logra colocando reguladores de flujo en las líneas de escape del pistón. Es importante mencionar que la estrangulación se debe realizar en la línea de escape, de lo contrario se perderá fuerza y se obtendrá un movimiento oscilante.

Como se desea que este movimiento retardado, se ejecute en un solo sentido, se utilizan reguladores unidireccionales como se ve en la Figura 4.10, esto significa que existirá estrangulación en un solo sentido, cuando el escape sea reducido. Si se desea restringir la

velocidad tanto de regreso como de salida, deben ponerse en ambos lados del pistón, esto asegura que el pistón tenga regulación tanto de entrada, como de salida.

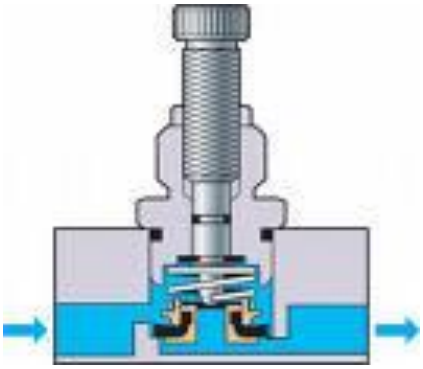


Figura 4. 10 Válvula reguladora de velocidad.

4.3.1.5 Electroválvulas.

Una electroválvula tiene dos partes fundamentales: el solenoide y la válvula. El solenoide convierte energía eléctrica en energía mecánica para que actúe la válvula.

Existen varios tipos de electroválvulas. En algunas electroválvulas el solenoide actúa directamente sobre la válvula proporcionando toda la energía necesaria para su movimiento. Es común que la válvula se mantenga cerrada por la acción de un muelle y que el solenoide la abra venciendo la fuerza del muelle (Figura 4.11). Esto quiere decir que el solenoide debe estar activado y consumiendo energía mientras la válvula deba estar abierta.

Figura 4. 11 Electroválvulas neumáticas.

4.3.2 Secuencia de la Bloquera.

El proceso de la bloquera consta de 4 pistones y dos motores, al presionar el pulsador de arranque el motor 1 que está acoplado a una banda permanecerá encendido por un tiempo hasta que el material llegue al depósito de presión, en donde el pistón A saldrá por tres ocasiones, dando forma al bloque, luego el pistón B saldrá para elevar el material elaborado; inmediatamente el pistón C saldrá para trasladarlo a una segunda banda, el pistón B y C retornan a su posición inicial, en este momento se prenderá el segundo motor y a la vez saldrá el pistón D; cuando el material llegue al final de la banda, el pistón D retornará llevando el material, y terminando la secuencia del proceso de la bloquera.

Todo lo dicho anteriormente se resume en la siguiente secuencia:

$$(A+A-)3 \rightarrow B+ \rightarrow C+ \rightarrow \frac{B-}{C-} \rightarrow D+ \rightarrow D-$$

Se debe mencionar que los pistones A, B y D son pistones de doble efecto, mientras que el pistón C es de simple efecto con retorno de resorte, el mismo que para su funcionamiento se debe enclavar la señal a la electroválvula.

A continuación se presenta el diagrama esquemático de la secuencia de la bloquera (Figura 4.12)

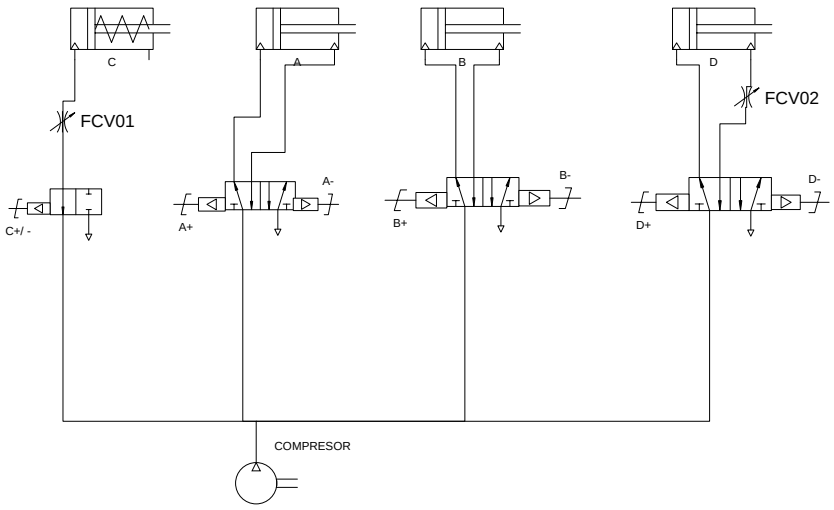


Figura 4. 12 Diagrama esquemático bloquera.

4.3.2.1 Diagrama espacio – fase.

El diagrama espacio – fase (Figura 4.13), es un diagrama donde se ve en el eje de las "x" el tiempo y en el eje de las "y" el desplazamiento de los pistones. En resumen se ve el desplazamiento de los pistones en función del tiempo.

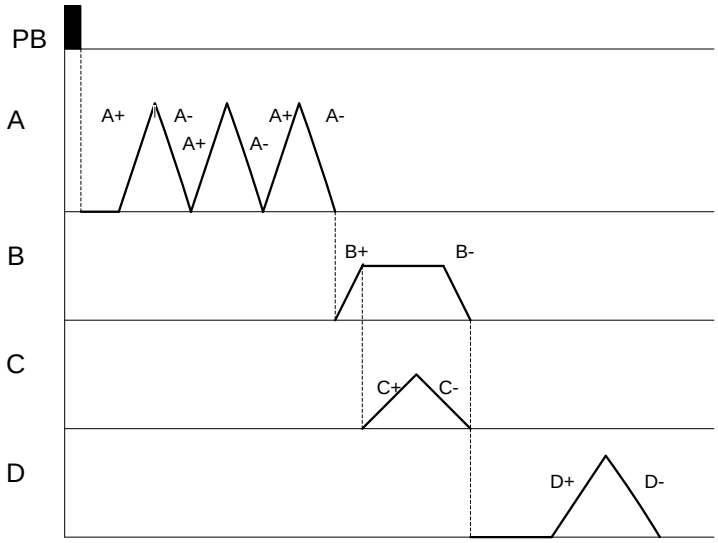


Figura 4. 13 Diagrama espacio – fase.

4.3.3 Programación del Proceso.

Para la programación del proceso se utilizará el software diseñado, el mismo que se detalla en el manual del PLC ByL, que se encuentra en los Anexos.

Para la ejecución del proceso en el software se necesitarán:

- 2 Entradas digitales, detalladas en la Tabla 4.2.

Entradas Físicas	Accionamiento
I1	Inicio
I2	Parada Total

Tabla 4. 2 Entradas digitales.

- 9 Salidas digitales, que se detallan a continuación en la Tabla 4.3.

Salidas Físicas	Accionamiento
Q1	Motor 1
Q2	A+
Q3	A-
Q4	B+
Q5	C+
Q6	B-
Q7	Motor 2
Q8	D+
Q9	D-

Tabla 4. 3 Salidas digitales.

- Timers, ya que el proceso no cuenta con finales de carrera, se lo ejecuta controlando tiempos.
- Contador.
- Set Reset.
- OR.
- XOR.
- AND.
- NOT.

En la Figura 4.14 se detalla el programa, en diagrama de bloques, el mismo que se utilizó para la programación del PLC.

4.3.4 Simulación del Proceso.

Una vez concluido la programación de bloques, se procede a realizar la simulación del mismo, para poder verificar si se cumple la secuencia y así realizar los respectivos cambios, si se requiere.

Para poder acceder a la simulación se da un clic en el icono de **simulación**, y aparecerá la Figura 4.15. Más detalles en el manual del PLC.

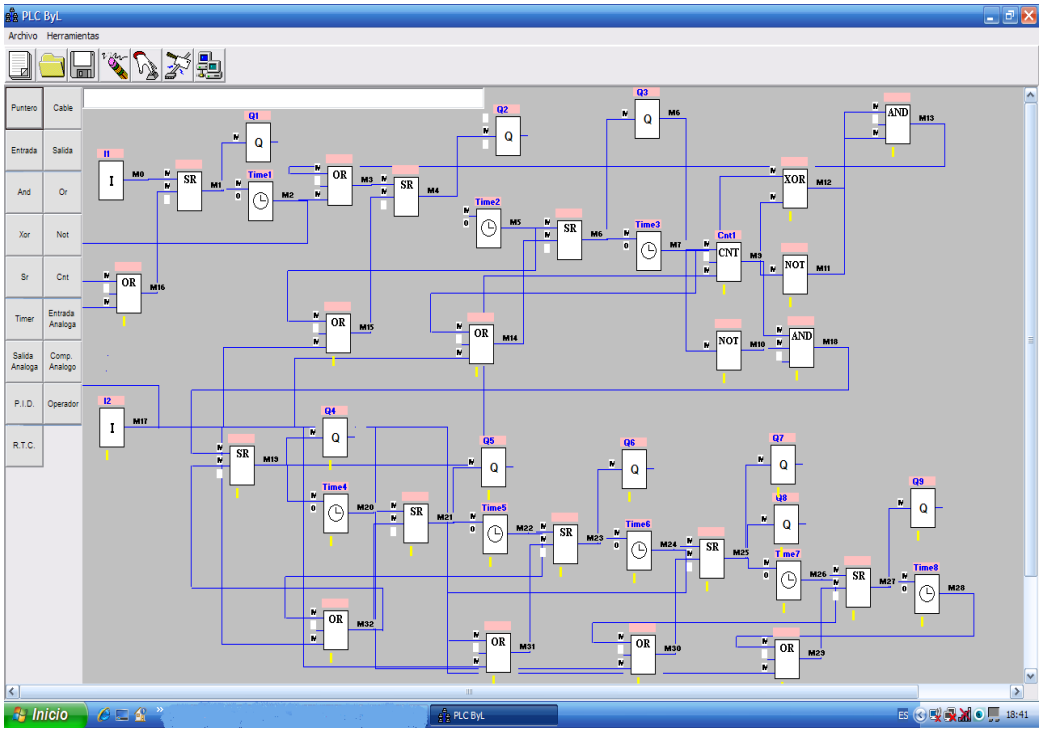


Figura 4. 14 Programación en bloques.

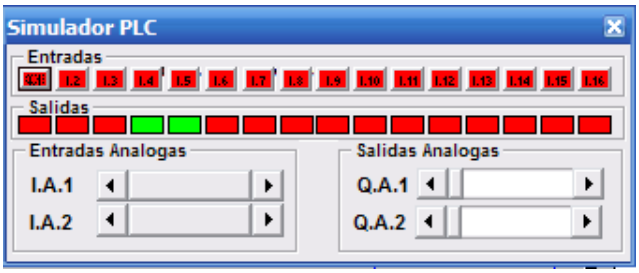


Figura 4. 15 Pantalla de simulación.

En la Figura 4.16, se observa el monitoreo en el LCD del PLC.

Figura 4. 16 Monitoreo del PLC.

En la Figura 4.17 se observa la implementación del PLC a la bloquera.

Figura 4. 17 Implementación del PLC.

En la siguiente Figura se presenta la secuencia de funcionamiento de la bloquera.

1. Inicio del proceso (Figura 4.18)

Figura 4. 18 Inicio del proceso.

2. Preparación de materia prima a materia elaborada (Figura 4.19)

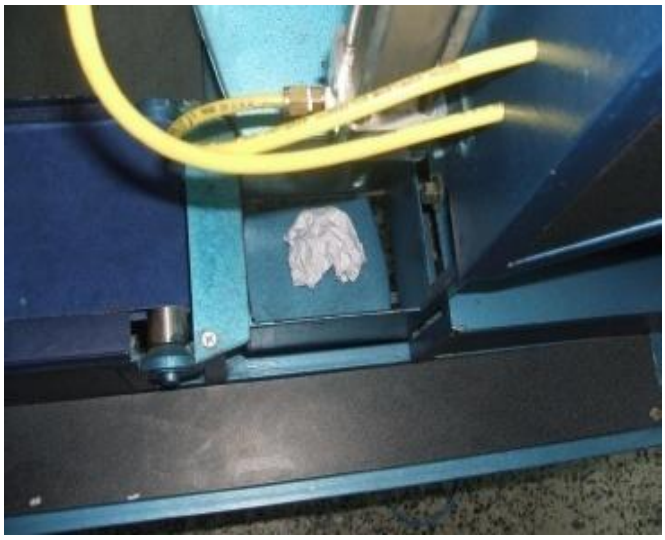


Figura 4. 19 Comprimido del material.

3. Salida del material elaborado y envío a la banda transportadora (Figura 4.20)

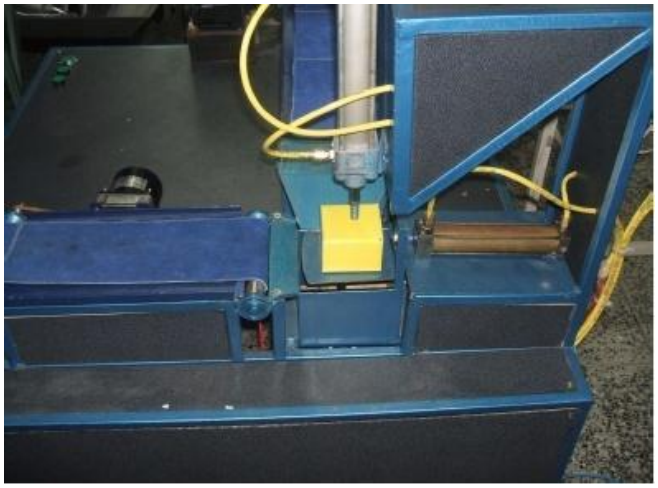


Figura 4. 20 Salida del material elaborado.

4. Al final de la banda transportadora el material elaborado es transportado, y así finaliza el proceso (Figura 4.21)

Figura 4. 21 Traslado del producto terminado.

4.4 ESTUDIO FINANCIERO.

4.4.1 Indicadores de Evaluación Financiera.

Los indicadores financieros indican la factibilidad de ejecución del proyecto, determinando parámetros de atractividad para el inversionista. Así se analizan los siguientes indicadores.

VAN: Valor Actual Neto

TIR: Tasa Interés de Retorno

PAY-BACK: Periodo de Recuperación de la Inversión

Primero se realizará el costo total para elaborar una sola placa, que vendrá siendo el costo de la materia prima, a continuación en la Tabla 4.4 se detallan los materiales y sus costos para elaborar un solo PLC.

ITEM	DESCRIPCION	CANTIDAD	V. UNITARIO USD	V.TOTAL USD
1	Pantalla LCD	1	35	35
2	Optoacoplador 4N35	32	0,35	11,2
3	Transistor 2N3904	1	1	1
4	TTL LS164N	10	1	10
5	Relés	16	0,8	12,8
6	Leds	17	0,8	13,6

7	Resistencias	78	0,04	3,12
8	Resistencias de montaje superficial	16	0,08	1,28
9	Borneras de 2 conectores	30	0,35	10,5
10	Resistencias de precisión	3	0,7	2,1
11	Reloj cristal interno	1	3	3
12	Pila	1	0,5	0,5
13	Conector DB9	1	1,2	1,2
14	Diodos	36	0,15	5,4
15	Max 232	1	1,5	1,5
16	PIC 18F4620	1	12	12
17	Cable serial	1	15	15
18	Conversor USB - Serial	1	30	30
19	Crital cuarzo 20MHz	1	0,8	0,8
20	Estaño			3
21	Cautin	1	1,5	1,5
22	Crema para soldar	1	0,9	0,9
23	Rooteada de placa	1	80	80
24	Envío			3
25	Quemador de PIC	1	25	25
26	Otros			41
27	Caja	1	10	10
			Valor Total	334,4

Tabla 4. 4 Costos para la elaboración de un PLC.

Para calcular los indicadores de evaluación se debe elaborar el flujo nominal de efectivo. Para el cálculo de pérdidas y ganancias, se tiene que realizar una pequeña encuesta en las pequeñas industrias de la zona y la principales centros educativos de la provincia, luego de esto se proyecta una venta de 20 PLC anuales lo mismos que tendrán un costo de 500 dólares.

El valor de 500 dólares se obtuvo luego de realizar comparaciones con PLC que tengan similares características al que se va a ofertar, es así que el PLC marca OPLC con 8 entradas digitales, 8 salidas digitales, 2 entradas análogas, comunicación RS 232 y pantalla de 2 líneas tiene un precio de 990,95 dólares (Proforma del OPLC en Anexo 5).

En la comparación se tomó los PLC más accesibles, ya que si se realizara comparaciones con autómatas más conocidos y por ende más costosos, su precio se eleva considerablemente; es así que un PLC Siemens de 8 entradas y 8 salidas digitales cuesta $725, el bloque de análogas $ 283.25, una fuente de 24 VDC $262, y por último el cable de conexión en $175, es decir estamos hablando alrededor de unos 1457 dólares (Proforma Siemens en Anexo 6).

Con lo anteriormente dicho se realiza la siguiente Tabla 4.5 del cálculo de flujo proyectado a cinco años con un 20% de crecimiento en ventas por cada año transcurrido.

DETALLE INGRESO	Años				
	2009	2010	2011	2012	2013
Ingreso	10.000,00	12.000,00	14.400,00	17.280,00	20.736,00
Total Ingreso	**10.000,00**	**12.000,00**	**14.400,00**	**17.280,00**	**20.736,00**
EGRESO					
Gasto de Materia Prima	6.680,00	6.876,39	7.078,56	7.286,67	7.500,90
Gasto Servicios Básicos	100,00	102,94	105,97	109,08	112,29
Otros Gastos	200,00	205,88	211,93	218,16	224,58
Total Egreso	**6.980,00**	**7.185,21**	**7.396,46**	**7.613,91**	**7.837,76**
Superávit / Déficit	**3.020,00**	**4.814,79**	**7.003,54**	**9.666,09**	**12.898,24**

Tabla 4. 5 Cálculo de flujo.

4.4.2 Valor Actual Neto VAN[29]

Este criterio plantea que el proyecto debe aceptarse si su valor net es igual o superior a cero, donde el VAN es la diferencia entre todos los ingresos y egresos expresados en

[29] Brealey R. Fundamentos de Financiación Empresarial. Ed. Mc Graw-Hill. España 1998. Pag. 78.

moneda actual. Además es el valor monetario que resulta de restar la suma de los flujos descartados a la inversión inicial; es decir, comparando todas las ganancias esperadas contra todos los desembolsos necesarios para producir esas ganancias en términos de su valor equivalente en ese momento o tiempo cero.

Es dado que para aceptar un proyecto las ganancias deberán ser mayores que los desembolsos, lo cual dará el resultado que el VAN sea mayor que cero.

La ecuación de cálculo utilizada para calcular el VAN es:

$$VAN = -Inversion + \frac{Flujo\ año1}{(1+i)^1} + \frac{Flujo\ año\ 2}{(1+i)^2} + \frac{Flujo\ año\ 3}{(1+i)^3} + \cdots \frac{Flujo\ año\ n}{(1+i)^n} \qquad \text{EC. 4.1}$$

Donde:

Flujo: Es el valor obtenido del flujo nominal de cada año

I: Es el costo de oportunidad del 23.03%

Inversión: En la Tabla 4.5 se obtiene que para la construcción de 20 PLC se tendría una inversión de $ 6680.

$$VAN = -Inversion + \frac{Flujo\ año1}{(1+i)^1} + \frac{Flujo\ año\ 2}{(1+i)^2} + \frac{Flujo\ año\ 3}{(1+i)^3} + \cdots \frac{Flujo\ año\ n}{(1+i)^n}$$

$$VAN = -6680 + \frac{3020}{(1+0.2303)^1} + \frac{4814.79}{(1+0.2303)^2} + \frac{7003.54}{(1+0.2303)^3} + \frac{9666.09}{(1+0.2303)^4} + \frac{12293.24}{(1+0.2303)^5}$$

VAN = - 6,680 + 2,454.68 + 3,180.93 + 3,760.83 + 4,218.97 + 4,575.88

VAN = $ 11,511.29

El valor actual neto del proyecto es de 11,511.29 dólares, lo que quiere decir que representa una decisión favorable para ejecutar el proyecto de inversión analizado.

4.4.3 Tasa Interna de Retorno[30].

El criterio de la tasa de retorno evalúa el proyecto en función de una única tasa de rendimiento por periodo, en el cual la totalidad de los beneficios actualizados son exactamente iguales a los desembolsos en moneda actual.

[30] Brealey R. Fundamentos de Financiación Empresarial. Ed. Mc Graw-Hill. España 1998. Pag 115.

Además representa la rentabilidad obtenida en proporción directa del capital invertido o también a las tasas de descuento que hace que el VPN sea 0

Se calcula utilizando un sistema de aproximaciones sucesivas, partiendo de una tasa de descuento inicial que se establece en forma subjetiva, la misma que se va ajustando hasta obtener dos tasas que den como resultado un VAN = 0 y luego por interpolación se calcula TIR

Sin embargo en la actualidad existen muchos sistemas de obtención de la TIR, que viabilizan el procedimiento, el que se utilizó en el presente proyecta a través del programa Excel.

Cuando el valor actual neto es igual a cero, entonces se calcula la tasa de interés de la fórmula del VAN, tasa que representa la Tasa Interna de Retorno, y es atractiva cuando es superior o igual al costo de oportunidad, es decir que en el análisis del presente proyecto debe ser mayor o igual al 23.03%

VAN = 0

De la ecuación 4.1 se tiene:

$$0 = -Inversion + \frac{Flujo\ año 1}{(1+i)^1} + \frac{Flujo\ año 2}{(1+i)^2} + \frac{Flujo\ año 3}{(1+i)^3} + \cdots \frac{Flujo\ año n}{(1+i)^n}$$

$$0 = -6680 + \frac{3020}{(1+i)^1} + +\frac{4814.79}{(1+i)^2}\ \frac{7003.54}{(1+i)^3} + \frac{9666.09}{(1+i)^4} + \frac{12293.24}{(1+i)^5}$$

Luego de realizar varias veces el cálculo, el TIR para el proyecto es del 75% el mismo que es superior a 23.03% del costo de oportunidad. Entonces el proyecto es favorable porque tiene un rendimiento efectivo de los recursos invertidos del proyecto.

4.4.4 Periodo de Recuperación.[31]

El período de recuperación (PR) indica el tiempo estimado en que se puede recuperar la inversión realizada, basándose en los flujos que genera en cada periodo de su vida útil, es un indicador más subjetivo pero sirve para evaluar un proyecto. El periodo de recuperación se calcula con la siguiente ecuación:

[31] http://www.pymesfuturo.com/pri.htm

$$PR = \frac{Flujo\ Neto * 24}{FN1 + FN\ 2} \qquad \text{Ec. 4.2}$$

$$PR = \frac{6680 * 24}{3020 + 4814}$$

PR = 20.45

PR= 1 año y 8 meses

Existe otra forma de calcular el periodo de recuperación más sencilla:

$$Período\ de\ Recuperación = \frac{Inversión\ Fija\ Inicial}{Flujo\ Efectivo\ Anual\ año\ 1} \qquad \text{Ec. 4.3}$$

$$Período\ de\ Recuperación = \frac{6680}{3020}$$

PR = 2.2

PR = 2 años y 2 meses

4.4.5 Análisis de Sensibilidad.

De acuerdo a los parámetros de evaluación analizados anteriormente se concluye que, el proyecto de construir un PLC con Microcontroladores es factible, así lo indica la Tabla 4.6

INDICADORES	EVALUACIÓN	DECISIÓN
PERIODO DE RECUPERCIÓN	2 AÑOS Y 2 MESES	FAVORABLE
TASA INTERNA DE RETORNO	75%	FAVORABLE
VALOR PRESENTE NETO	11,511.29	FAVORABLE

Tabla 4. 6 Análisis de sensibilidad.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES.

Una vez finalizado el proyecto de tesis "Diseño y Construcción de un PLC en base de Microcontroladores", se obtienen las siguientes conclusiones:

5.1 CONCLUSIONES.

- Se logró el objetivo principal que fue trazado, ya que se pudo construir un PLC que cumpliera con las características básicas que posee un autómata.

- Ya que dispone de un algoritmo de control PID, se puede realizar controles de nivel, caudal, temperatura, etc.

- El software del PLC es fácil de entender, ya que es similar a otros que existen en el mercado, pues se trata de un lenguaje muy conocido, como es el caso de la programación por bloques, y usado a menudo en la programación de PLCs.

- Como se trata de un prototipo no cumple con ninguna norma estandarizada, y se tomó como referencia para el diseño en autómatas existentes en el mercado (PLC Siemens LOGO).

- Cuenta con un LCD gráfico para el monitoreo de las entradas y salidas, y permite observar el estado del PLC.

- El PLC tiene entradas digitales, análogas y sus respectivas salidas para el manejo de procesos de mediana y baja complejidad.

- Cada salida dispone de un led de visualización, además existe un led para el RTC.

- Para el diseño del software se utilizó el programa Visual Basic versión 6.3, la misma que es muy fácil de encontrar y no necesita de licencias especiales.

- El software del PLC es un archivo auto ejecutable, por lo que no se necesita tener instalado Visual Basic para su ejecución; y además es un archivo que ocupa poco espacio, por lo se adapta fácilmente a cualquier computadora del medio.

- Se implementó el PLC en una sola tarjeta, esto facilita su construcción en grandes cantidades, así como su montaje e implementación. Además lo hace más compacto y fácil de manipular. (Diagrama esquemático de la tarjeta del PLC Anexo 7)

- Los elementos utilizados en el proyecto son fáciles de adquirir en el mercado nacional, y sus costos no son extremadamente caros, lo que permitió abaratar los costos del mismo.

- El proyecto implementado se enfoca a medianas y pequeñas industrias del país, y sobre todo se lo puede utilizar como material didáctico en Universidades, debido a su bajo costo.

- Se determinó que el Software de Programación Visual Basic, es una poderosa herramienta que no sólo debe ser empleada para realizar proyectos en el área de Instrumentación Virtual, sino también se pueden desarrollar proyectos de monitoreo y control de procesos en general.

5.2 RECOMENDACIONES.

- Para el correcto funcionamiento del PLC, se debe leer detenidamente el manual que posee el PLC y que se encuentra en los Anexos.

- Como se especifica en el manual del PLC, los voltajes recomendados para el normal funcionamiento son para entradas digitales de 10 a 12V DC, para las entradas análogas de 0 a 10 V DC y la alimentación es de 110VAC a 12VDC.

- Para la comunicación con la PC, el PLC consta de un puerto serial, pero debido a los avances tecnológicos, este puerto está desapareciendo, lo cual no es un problema ya que existen diferentes tipos de conversores en el mercado, fáciles de adquirir, con los que el autómata funciona correctamente previa la configuración del puerto USB.

- La instalación del software del PLC, es muy sencillo ya que no se requiere que la PC tenga características especiales, tan solo se debe copiar el archivo autoejecutable en el disco C: de la PC, y no es importante la versión del sistema operativo de Windows.

- Se debe tener el mismo cuidado en la manipulación del PLC, como se lo haría con cualquier otro del mercado. Si la pantalla del LCD del PLC está en modo de espera, no se debe realizar ninguna otra acción ya que esto provocaría un mal funcionamiento del PLC.

- Para el aumento de bloque de entradas y salidas, se debe modificar la programación tanto de Visual Basic, como en MikroBasic; y comprobar si el PIC cuenta con puertos y memoria libre, de lo contrario se debe utilizar otro PIC en cascada.

- Se recomienda tomar en cuenta el diseño, tanto del software como del hardware, para en futuras modificaciones lograr perfeccionar el PLC; ya que se podría reducir el tamaño y mejorar sus características técnicas.

- Antes de construir una PCB, será necesario simular el diseño del circuito a través de la herramienta de simulación PROTEUS y armar el mismo en un protoboard, para no tener inconvenientes tanto económicos, como de tiempo en un mal diseño.

ANEXOS

ANEXOS.

Anexo 1: Placas del PLC ByL.

Anexo 2: Programación en Visual Basic.

Anexo 3: Programación en MicroCode.

Anexo 4: Hoja de Características del GLCD.

Anexo 5: Proforma del PLC Unitronics V350

Anexo 6: Hoja de Características y Proforma del PLC S7-200 CPU 224XP

Anexo 7: Diagrama esquemático de las Tarjetas del PLC.

ANEXO 3	Programación en MicroCode Studio	1

```
INCLUDE "MODEDEFS.BAS"
  adcon1=7
P1 VAR PORTB.0
P2 VAR PORTB.1
P3 VAR PORTB.2
P4 VAR PORTB.3
P5 var PORTB.4
P6 VAR PORTB.5
P7 VAR PORTB.6
P8 VAR PORTB.7
P9 VAR PORTA.2
P10 VAR PORTA.3
P11 VAR PORTC.1
RE VAR BYTE:DATO VAR BYTE
A VAR BYTE:B VAR BYTE:C VAR BYTE:D VAR BYTE
E VAR BYTE:F VAR BYTE:G VAR BYTE:H VAR BYTE
I VAR BYTE:J VAR BYTE:K VAR BYTE:L VAR BYTE
M VAR BYTE:N VAR BYTE:NN VAR BYTE:O VAR BYTE
P VAR BYTE:Q VAR BYTE:R VAR BYTE:S VAR BYTE
T VAR BYTE:U VAR BYTE:V VAR BYTE:W VAR BYTE
X VAR BYTE:Y VAR BYTE:Z VAR BYTE:AA VAR BYTE
```

```
INICIO:
   IF P1=1 THEN A=1: GOSUB ENVIO:GOSUB DOS
   IF P2=1 THEN E=1:GOSUB ENVIO1:GOSUB DOS
   IF P3=1 THEN I=1:GOSUB ENVIO2:GOSUB DOS
   IF P4=1 THEN M=1:GOSUB ENVIO3:GOSUB DOS
   IF P9=1 THEN P=1:GOSUB ENVIO4:GOSUB DOS
   IF P10=1 THEN T=1:GOSUB ENVIO5:GOSUB DOS
   IF P11=1 THEN X=1:GOSUB ENVIO6:GOSUB DOS
   SERIN PORTA.1,N2400,30,INICIO,["P"],DATO
   IF DATO ="1" THEN HIGH PORTA.7:PAUSE 2000:LOW PORTC.0
 GOTO INICIO


DOS:
   IF P1=1 THEN B=1: GOSUB ENVIO:GOSUB TRES
   IF P2=1 THEN F=1:GOSUB ENVIO1:GOSUB TRES
   IF P3=1 THEN J=1:GOSUB ENVIO2:GOSUB TRES
   IF P4=1 THEN N=1:GOSUB ENVIO3:GOSUB TRES
   IF P9=1 THEN Q=1:GOSUB ENVIO4:GOSUB TRES
   IF P10=1 THEN U=1:GOSUB ENVIO5:GOSUB TRES
   IF P11=1 THEN Y=1:GOSUB ENVIO6:GOSUB TRES
 GOTO DOS
```

```
TRES:
  IF P1=1 THEN C=1: GOSUB ENVIO:GOSUB CUATRO
  IF P2=1 THEN G=1:GOSUB ENVIO1:GOSUB CUATRO
  IF P3=1 THEN K=1:GOSUB ENVIO2:GOSUB CUATRO
  IF P4=1 THEN NN=1:GOSUB ENVIO3:GOSUB CUATRO
  IF P9=1 THEN R=1:GOSUB ENVIO4:GOSUB CUATRO
  IF P10=1 THEN V=1:GOSUB ENVIO5:GOSUB CUATRO
  IF P11=1 THEN Z=1:GOSUB ENVIO6:GOSUB CUATRO
 GOTO TRES

CUATRO
  IF P1=1 THEN D=1:GOSUB ENVIO: GOSUB INICIO
  IF P2=1 THEN H=1:GOSUB ENVIO1:GOSUB INICIO
  IF P3=1 THEN L=1:GOSUB ENVIO2:GOSUB INICIO
  IF P4=1 THEN O=1:GOSUB ENVIO3:GOSUB INICIO
  IF P9=1 THEN S=1:GOSUB ENVIO4:GOSUB INICIO
  IF P10=1 THEN W=1:GOSUB ENVIO5:GOSUB INICIO
  IF P11=1 THEN AA=1:GOSUB ENVIO6:GOSUB INICIO
 GOTO CUATRO
ENVIO:
 IF P5=1 AND A=1 THEN
  FOR RE=0 TO 50:SEROUT PORTA.0,N2400,["CA1"]:PAUSE
5:NEXT:a=0:RETURN
 ENDIF
 IF P7=1 AND C=1 THEN
```

```
FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["CA3"]:PAUSE 5:NEXT:C=0:RETURN
 ENDIF
 IF P8=1 AND D=1 THEN
  FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["CA4"]:PAUSE 5:NEXT:D=0:RETURN
 ENDIF
 GOTO ENVIO
ENVIO1:
 IF P5=1 AND E=1 THEN
  FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["CC1"]:PAUSE 5:NEXT:E=0:RETURN
 ENDIF
 IF P6=1 AND F=1 THEN
  FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["CC2"]:PAUSE 5:NEXT:F=0:RETURN
 ENDIF
 IF P7=1 AND G=1 THEN
  FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["CC3"]:PAUSE 5:NEXT:G=0:RETURN
 ENDIF
 IF P8=1 AND H=1 THEN
  FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["CC4"]:PAUSE 5:NEXT:H=0:RETURN
 ENDIF
 GOTO ENVIO1
```

```
ENVIO2:
  IF P5=1 AND I=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["L1"]:PAUSE
5:NEXT:I=0:RETURN
  ENDIF
  IF P6=1 AND J=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["L2"]:PAUSE
5:NEXT:J=0:RETURN
  ENDIF
  IF P7=1 AND K=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["L3"]:PAUSE
5:NEXT:K=0:RETURN
  ENDIF
  IF P8=1 AND L=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["L4"]:PAUSE
5:NEXT:L=0:RETURN
  ENDIF
 GOTO ENVIO2
ENVIO3:
  IF P5=1 AND M=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["S1"]:PAUSE
5:NEXT:M=0:RETURN
  ENDIF
  IF P6=1 AND N=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["S2"]:PAUSE
5:NEXT:N=0:RETURN
  ENDIF
  IF P7=1 AND NN=1 THEN
   FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["S3"]:PAUSE
5:NEXT:O=0:RETURN
  ENDIF
  IF P8=1 AND O=1 THEN
```

```
 FOR RE=0 TO 10:SEROUT PORTA.0,N2400,["S4"]:PAUSE 5:NEXT:P=0:RETURN
 ENDIF
GOTO ENVIO3
```

- **Recepción de Datos (PIC 18F4620).**

```
INCLUDE "MODEDEFS.BAS"
 adcon1=7
  '**********  perifericos **********
 LED1 VAR PORTB.0
 LED2 VAR PORTB.1
 LED3 VAR PORTB.2
 LED4 VAR PORTB.3
 LED5 VAR PORTA.2
LED6 VAR PORTA.3
 LED7 VAR PORTB.5
 P1 VAR PORTB.4
 P3 VAR PORTB.6
CAB VAR BYTE:CC VAR BYTE:IN VAR BYTE
 CE VAR BYTE: LI VAR BYTE:CA VAR BYTE
 SC VAR BYTE:SA VAR BYTE:CJ1 var byte
 CJ2 VAR BYTE:CJ3 VAR BYTE:R VAR BYTE
 CJ4 VAR BYTE:RE VAR BYTE:AS VAR BYTE:S VAR BYTE
 A VAR BYTE:B VAR BYTE:C VAR BYTE:D VAR BYTE
 E VAR BYTE:F VAR BYTE:G VAR BYTE:H VAR BYTE
```

```
INICIO:
  IF P1=1 THEN CAB=0:GOSUB ENLACE
  IF P1=0 AND CAB=0 THEN CAB=1:GOSUB ENLACE

  IF P3=1 THEN CC=0:GOSUB ENLACE
  IF P3=0 AND CC=0 THEN CC=1:GOSUB ENLACE

 IF PORTB.0=1 AND PORTB.1=1 AND PORTB.2=1 AND PORTB.3=1 AND PORTA.3=1 THEN
  FOR RE=0 TO 50:SEROUT PORTA.1,N2400,["P1"]:PAUSE 5:NEXT:PAUSE 1000:LOW LED6
 ENDIF
  FOR R=0 TO 300
   SERIN PORTA.0,N2400,5,DOS,["CA"],IN :GOSUB ENLACE
   DOS:
   SERIN PORTA.0,N2400,3,TRES,["CC"],CE:GOSUB ENLACE
   TRES:
   SERIN PORTA.0,N2400,3,CUATRO,["L"],LI:GOSUB ENLACE
   CUATRO:
   SERIN PORTA.0,N2400,3,CINCO,["AB"],CA:GOSUB ENLACE
   CINCO:
   SERIN PORTA.0,N2400,3,SEIS,["CD"],SC:GOSUB ENLACE
   SEIS:
   SERIN PORTA.0,N2400,3,SIETE,["CS"],AS:GOSUB ENLACE
   SIETE:
   SERIN PORTA.0,N2400,3,INICIO,["S"],SA:HIGH LED6:GOSUB ENLACE
```

```
 PAUSE 2
 NEXT
GOTO INICIO
ENLACE:

 IF IN="1" AND CAB=1 THEN A=1:CJ1=1:HIGH LED1
 IF IN="1" AND CAB=0 THEN A=2:CJ1=1:LOW LED1
 IF IN="2" AND CAB=1 THEN B=1:CJ2=1:HIGH LED2
 IF IN="2" AND CAB=0 THEN B=2:CJ2=1:LOW LED2
 IF IN="3" AND CAB=1 THEN C=1:CJ3=1:HIGH LED3
 IF IN="3" AND CAB=0 THEN C=2:CJ3=1:LOW LED3
 IF IN="4" AND CAB=1 THEN D=1:CJ4=1:HIGH LED4
 IF IN="4" AND CAB=0 THEN D=2:CJ4=1:LOW LED4

 IF CE="1" AND CAB=1 THEN A=2:CJ1=1:HIGH LED1
 IF CE="1" AND CAB=0 THEN A=1:CJ1=1:LOW LED1
 IF CE="2" AND CAB=1 THEN B=2:CJ2=1:HIGH LED2
 IF CE="2" AND CAB=0 THEN B=1:CJ2=1:LOW LED2
 IF CE="3" AND CAB=1 THEN C=2:CJ3=1:HIGH LED3
 IF CE="3" AND CAB=0 THEN C=1:CJ3=1:LOW LED3
 IF CE="4" AND CAB=1 THEN D=2:CJ4=1:HIGH LED4
 IF CE="4" AND CAB=0 THEN D=1:CJ4=1:LOW LED4
 IF SA="1" THEN A=4:CJ1=1:HIGH LED1
 IF SA="2" THEN B=4:CJ2=1:HIGH LED2
 IF SA="3" THEN C=4:CJ3=1:HIGH LED3
```

```
IF CJ2=1 AND B=1 THEN N=1
   IF CJ2=1 AND B=2 THEN N=2
   IF CJ2=1 AND F=1 THEN N=3
   IF CJ2=1 AND F=2 THEN N=4
   IF CJ2=1 AND B=3 THEN N=5
  IF CJ3=1 AND C=1 THEN O=1
   IF CJ3=1 AND C=2 THEN O=2
   IF CJ3=1 AND G=1 THEN O=3
   IF CJ3=1 AND G=2 THEN O=4
   IF CJ3=1 AND C=3 THEN O=5
   IF CJ4=1 AND D=4 THEN P=1

 GOSUB SIMULAR
 RETURN
SIMULAR:

  IF ((M=1 OR M=3) AND (N=3 OR N=1 OR N=5) AND O=5 AND P=1) THEN
LOW LED5:S=0
  IF ((M=1 OR M=3) AND (N=2 OR N=4 OR N=5) AND O=5 AND P=1) THEN
LOW LED5:S=0
  IF ((M=2 OR M=4OR M=6) AND (N=1 OR N=3) AND O=5 AND P=1) THEN
LOW LED5:S=0
IF ((M=2 OR M=4OR M=6) AND (N=2 OR N=4 OR N=5) AND O=5 AND P=1)
THEN HIGH          LED5:S=1

RETURN
```

Form1.

```
Private Sub Form_Load()
   Estado1 = True
End Sub
```

Form2.

```
Private Sub Command1_Click(Index As Integer)
a = Index
If a = 0 Then
   Cable = False

   For i = 1 To Contador
      For j = 1 To ContBorrado
         If BorrarFuncion(j, 0) = i And BorrarFuncion(j, 3) = 1 Then
            condicion = True
         End If
      Next j
      If condicion = False Then
         Form3.Picture2(i).DragMode = 1
      End If
      condicion = False
   Next i

End If
If a = 1 Then
   For i = 1 To Contador
```

```
Form3.Picture2(i).DragMode = 0
   Next i
   Cable = True
ElseIf a > 1 Then
   If Cable = False Then
      If PressHerr = False Then
         PressHerr = True
      End If
   End If
End If
End Sub
Private Sub Command1_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
Select Case Index
   Case 0
   Command1(Index).ToolTipText = "Puntero"
   Case 1
   Command1(Index).ToolTipText = "Cable"
   Case 2
   Command1(Index).ToolTipText = "Entradas"
   Case 3
   Command1(Index).ToolTipText = "Salidas"
   Case 4
   Command1(Index).ToolTipText = "And"
   Case 5
   Command1(Index).ToolTipText = "Or"
```

```
Case 6
   Command1(Index).ToolTipText = "Xor"
   Case 7
   Command1(Index).ToolTipText = "Not"
   Case 8
   Command1(Index).ToolTipText = "SR"
   Case 9
   Command1(Index).ToolTipText = "Cnt"
   Case 10
   Command1(Index).ToolTipText = "Timer"
   Case 11
   Command1(Index).ToolTipText = "Entrada Analoga"
   Case 12
   Command1(Index).ToolTipText = "Salida Analoga"
   Case 13
   Command1(Index).ToolTipText = "Comparador Analogo"
   Case 14
   Command1(Index).ToolTipText = "P.I.D"
   Case 15
   Command1(Index).ToolTipText = "Operador Matematico"
   Case 16
   Command1(Index).ToolTipText = "R.T.C"
End Select
End Sub
```

```
Private Sub Form_Load()
Form2.Left = 0
Form2.Top = 0
Form2.Show
Command1(0).Left = 0
Command1(0).Top = 0
For i = 1 To 16
   Load Command1(i)
   Command1(i).Left = i * 710 - 2 * 710 * Int(i / 2)
   Command1(i).Top = 700 * Int(i / 2)
   Command1(i).Visible = True
Next i
Command1(0).Caption = "Puntero"
Command1(1).Caption = "Cable"
Command1(2).Caption = "Entrada"
Command1(3).Caption = "Salida"
Command1(4).Caption = "And"
Command1(5).Caption = "Or"
Command1(6).Caption = "Xor"
Command1(7).Caption = "Not"
Command1(8).Caption = "Sr"
Command1(9).Caption = "Cnt"
Command1(10).Caption = "Timer"
Command1(11).Caption = "Entrada Analoga"
Command1(12).Caption = "Salida Analoga"
```

```
Command1(13).Caption = "Comp. Analogo"
Command1(14).Caption = "P.I.D."
Command1(15).Caption = "Operador"
Command1(16).Caption = "R.T.C."
End Sub
```

Form3.

```
Private Sub Picture1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)

Source.Move X - ValorX, Y - ValorY
If Form3.Label1(Indice).Caption <> "" Or Form3.Label2(Indice).Caption <> "" Or Form3.Label3(Indice).Caption <> "" Then
   MovLineas(1) = Trim(Mid(Form3.Label1(Indice).Caption, 2, 3)) 'Algoritmo para mover cables
   MovLineas(2) = Trim(Mid(Form3.Label2(Indice).Caption, 2, 3))
   MovLineas(3) = Trim(Mid(Form3.Label3(Indice).Caption, 2, 3))
   MovLineas(4) = Trim(Mid(Form3.Label4(Indice).Caption, 2, 3))
   For j = 1 To 4
      fgh = MovLineas(j)
      If fgh <> "" Then
         For i = 1 To ContLineas
            Color = i
            If fgh = EstadoCable(i) Then
               If j <> 4 Then
                  If Line3(i).X1 - Line3(i).X2 < 0 Then
                     Line3(i).X2 = Picture2(Index).Left
```

```
Else
                    Line5(i).X2 = Picture2(Index).Left
                End If
            End If
            Picture3.Line (Line2(i).X1, Line2(i).Y1)-(Line2(i).X2, Line2(i).Y2),
RGB(0, 0, 255)
            Select Case j
                Case 1
                    If Line3(i).X1 - Line3(i).X2 < 0 Then
                        Line3(i).Y1 = Picture2(Index).Top + 160
                    Else
                        Line5(i).Y1 = Picture2(Index).Top + 160
                    End If
                Case 2
                    If Line3(i).X1 - Line3(i).X2 < 0 Then
                        Line3(i).Y1 = Picture2(Index).Top + 375
                    Else
                        Line5(i).Y1 = Picture2(Index).Top + 375
                    End If
                Case 3
If Line3(i).X1 - Line3(i).X2 < 0 Then
                        Line3(i).Y1 = Picture2(Index).Top + 600
                    Else
                        Line5(i).Y1 = Picture2(Index).Top + 600
                    End If
                Case 4
```

```
If Line3(i).X1 - Line3(i).X2 < 0 Then
                    Line1(i).X1 = Picture2(Index).Left + 710
                    distline = Line3(i).X2 - Line1(i).X1
                    disty = Line3(i).Y1 - Line1(i).Y1
                    Line1(i).Y1 = Picture2(Index).Top + 355
                    Line1(i).Y2 = Picture2(Index).Top + 355
                Else
                    Line1(i).X1 = Picture2(Index).Left + 710
                    distline = Line5(i).X2 - Line1(i).X1
                    disty = Line5(i).Y1 - Line1(i).Y1
                    Line1(i).Y1 = Picture2(Index).Top + 355
                    Line1(i).Y2 = Picture2(Index).Top + 355
                End If
Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
  aa = Picture3.Point(X, Y)
  If aa <= 254 And Mover = False Then
    disty = Line3(aa).Y1 - Line1(aa).Y1
    Shape2.Left = Line2(aa).X1 - 50
    Shape2.Top = Line1(aa).Y1 + disty / 2
    Mover = True
  Else
    Mover = False
    Mover1 = False
  End If
```

```
 If PressHerr = True Then
   If ContRtc >= 2 Or (ContEnt >= 2 And a = 11) Or (ContTime >= 16 And a = 10) Or
(ContCnt >= 10 And a = 9) Or (ContQA >= 2 And a = 12) Or ContPID >= 1 Then
      Select Case a
         Case 11
            MsgBox "No puede colocar mas de 2 entradas analogicas", vbCritical
         Case 10
            MsgBox "No puede colocar mas de 16 Timers", vbCritical
         Case 9
            MsgBox "No puede colocar mas de 10 Contadores", vbCritical
         Case 16
            MsgBox "No puede colocar mas de 2 RTC", vbCritical
         Case 12
            MsgBox "No puede colocar mas de 2 Salidas Analogicas", vbCritical
         Case 14
            MsgBox "No puede colocar mas de 1 control PID", vbCritical
      End Select
      Exit Sub
   Else
      Contador = Contador + 1
      aaaaa = Contador
   End If
For j = 1 To ContBorrado
      If BorrarFuncion(j, 3) = 1 Then
         aaaaa = Val(BorrarFuncion(j, 0))
         indicefuncion = BorrarFuncion(j, 2)
```

```
 Private Sub Timer1_Timer(Index As Integer)
If TimeEstado(Index) = False And hhh(Index) >= Tiempos(Index) Then
   Timer1(Index).Interval = 0
   If TipoTimer(Index) = 0 Then
      salida1(Index) = True
      For u = 1 To Contador
         Marca1 = False
         Call AnalizarFuncion(u, Bloques(u))
      Next u
      For u = 1 To Contador
         Marca1 = False
         Call AnalizarFuncion(u, Bloques(u))
      Next u
Else
      For u = 1 To Contador
         Marca1 = False
         Call AnalizarFuncion(u, Bloques(u))
      Next u
   End If
End If
      For u = 1 To Contador
         Marca1 = False
         Call AnalizarFuncion(u, Bloques(u))
      Next u
End Sub
```

```
 End If
  End If
      For u = 1 To Contador
         Marca1 = False
         Call AnalizarFuncion(u, Bloques(u))
      Next u
End Sub
```

Form 4.

```
Private Sub Command1_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
If Botones(Index + 1) = False Then
   If Form5.Label2(Index).Caption = "Pulsador" Then
      Command1(Index).BackColor = RGB(0, 255, 0)
      Botones(Index + 1) = True
   End If
Else
   If Form5.Label2(Index).Caption = "Pulsador" Then
      Command1(Index).BackColor = RGB(255, 0, 0)
      Botones(Index + 1) = False
   End If
End If
For u = 1 To Contador
   Marca1 = False
   Call AnalizarFuncion(u, Bloques(u))
Next u
If Botones(Index + 1) = True Then
```

```
 Botones(Index + 1) = False
   End If
End If
Form5.Hide
Hg = True
End Sub
Private Sub Form_Load()
For i = 1 To 15
   Load Command1(i)
   Load Shape1(i)
   Command1(i).Left = Command1(i - 1).Left + Command1(i).Width + 34
   Shape1(i).Left = Shape1(i - 1).Left + Shape1(i).Width + 34
   Command1(i).Caption = "I" & "." & i + 1
   Command1(i).Visible = True
   Shape1(i).Visible = True
   Botones(i) = False
Next i
   Botones(16) = False
If ContComp > 0 Then
      HScroll1.Min = Comparadores(1, 0)
      HScroll1.Max = Comparadores(1, 1)
      HScroll2.Min = Comparadores(1, 0)
      HScroll2.Max = Comparadores(1, 1)
   Else
      HScroll1.Min = Operador(1, 0)
```

```
 HScroll1.Max = Operador(1, 1)
      HScroll2.Min = Operador(1, 0)
      HScroll2.Max = Operador(1, 1)
   End If
   For u = 1 To Contador
      hj1 = Bloques(u)
      If hj1 <> "R.T.C." Then
         Marca1 = False
         Call AnalizarFuncion(u, Bloques(u))
      Else
         ds = Val(Mid(Trim(Form3.Label5(u).Caption), 4, 1))
         RealTime(ds, 0) = u
         RealTime(ds, 1) = Mid(Trim(Form3.Label4(u).Caption), 2, 1)
         RealTime(ds, 2) = ds
      End If
   Next u
End Sub
Private Sub HScroll1_Change()
   Form3.Label1(Consulta1(1)).Caption = HScroll1.Value
   For u = 1 To Contador
      Marca1 = False
      Call AnalizarFuncion(u, Bloques(u))
   Next u
End Sub
```

Form 5.

```
Private Sub Combo1_click()
 Combo1.Visible = False
 Label2(Indice1).Caption = Combo1.Text
End Sub
Private Sub Form_Load()
For i = 1 To 15
   Load Label1(i)
   Load Label2(i)
   Label1(i).Left = Label1(0).Left
   Label2(i).Left = Label2(0).Left
   Label1(i).Top = Label1(i - 1).Top + Label1(i - 1).Height
   Label2(i).Top = Label2(i - 1).Top + Label2(i - 1).Height
   Label1(i).Caption = "I." & Format(i + 1, 0#)
   Label2(i).Caption = "Pulsador"
   Label1(i).Visible = True
   Label2(i).Visible = True
Next i
Label1(0).Caption = "I." & "1"
Label2(0).Caption = "Pulsador"
End Sub
Private Sub Label2_Click(Index As Integer)
   Combo1.Visible = True
   Combo1.Top = Label2(Index).Top
   Combo1.Left = Label2(Index).Left
```

```
 Indice1 = Index
End Sub
```

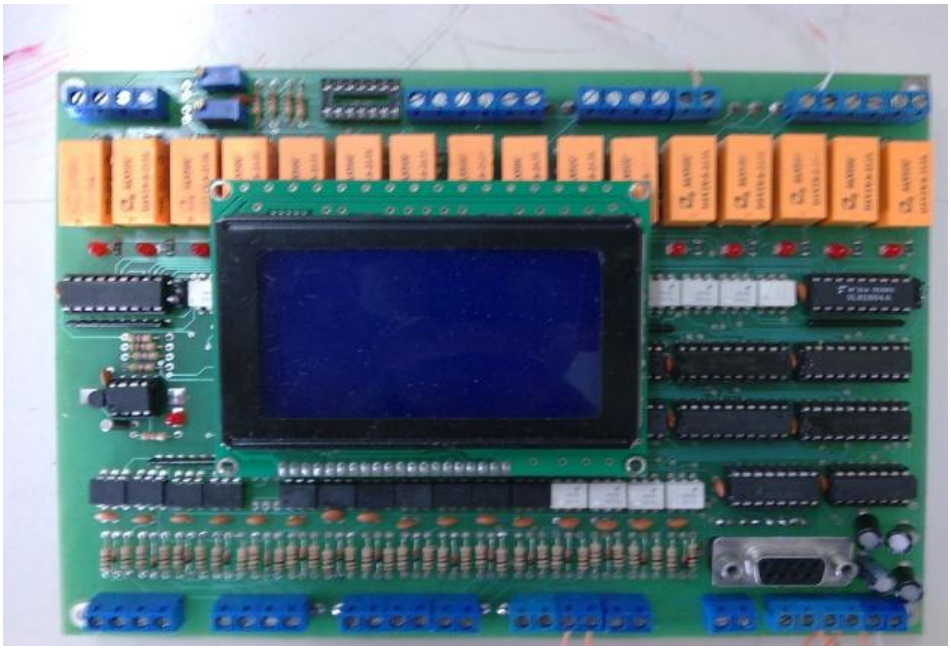
Form 6.

```
Private Sub Command1_Click()
Select Case Mid(Form6.Caption, 1, 3)
   Case "Cnt"
      ds = Val(Mid(Form6.Caption, 4, 2))
      Cnt(ds) = Val(Text1.Text)
   Case "Tim"
      ds = Val(Mid(Form6.Caption, 5, 2))
      Tiempos(ds) = Val(Combo1.Text) * 3600 + Val(Combo2.Text) * 60 + Val(Combo3.Text)
      If Option1(0).Value = True Then
         TipoTimer(ds) = 0
      Else
         TipoTimer(ds) = 1
      End If
   Case "Com"
      ds = Val(Mid(Form6.Caption, 5, 2))
      Comparadores(ds, 0) = Val(Text2.Text)
      Comparadores(ds, 1) = Val(Text3.Text)
      Comparadores(ds, 2) = Combo4.Text
      Comparadores(ds, 3) = Combo5.Text
      Comparadores(ds, 4) = Val(Text4.Text)
      Comparadores(ds, 5) = Val(Text5.Text)
      For i = 1 To 4
```

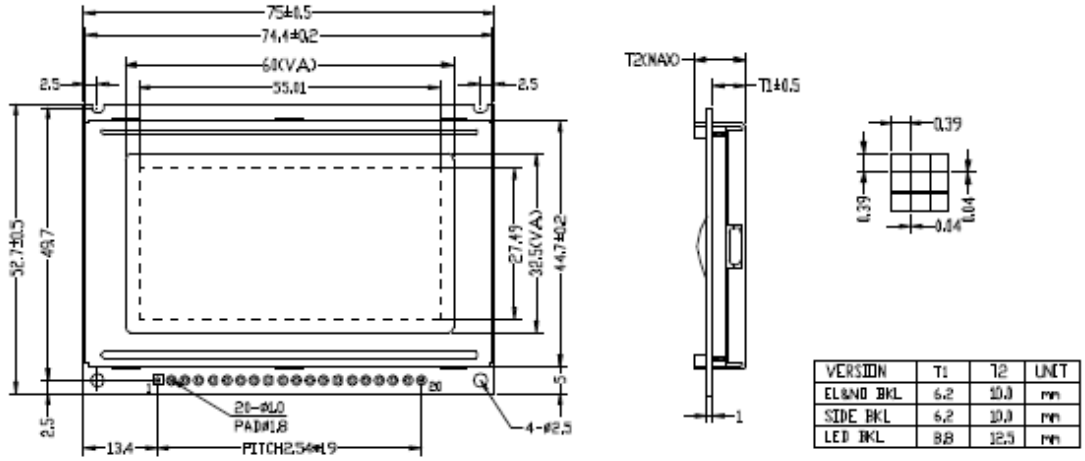
```
Comparadores(i, 0) = Val(Text2.Text)
        Comparadores(i, 1) = Val(Text3.Text)
        Operador(i, 0) = Val(Text2.Text)
        Operador(i, 1) = Val(Text3.Text)
    Next i
  Case "Ope"
    ds = Val(Mid(Form6.Caption, 5, 2))
    Operador(ds, 0) = Val(Text6.Text)
    Operador(ds, 1) = Val(Text7.Text)
    Operador(ds, 2) = Combo6.Text
    Operador(ds, 3) = Check1.Value
    Operador(ds, 4) = Val(Text8.Text)
    For i = 1 To 4
        Comparadores(i, 0) = Val(Text6.Text)
        Comparadores(i, 1) = Val(Text7.Text)
        Operador(i, 0) = Val(Text6.Text)
        Operador(i, 1) = Val(Text7.Text)
    Next i
Case "Pue"
    If Combo1.Text <> "Puertos" Then
        Open App.Path & "\Port.Port" For Random As #1 Len = 3
        Puertos.Estado = "S"
        Put #1, 1, Puertos
        Puertos.Com = Val(Trim(Mid(Combo7.Text, 4, 3)))
        Put #1, 2, Puertos
```

```
Close #1
    End If
  Case "RTC"
    ds = Val(Mid(Form6.Caption, 4, 2))
    RTC(ds, 1) = CDbl(DTPICker1.Value)
    RTC(ds, 2) = CDbl(DTPICker2.Value)
    RTC(ds, 3) = CDbl(DTPICker3.Value)
    RTC(ds, 4) = CDbl(DTPICker4.Value)
  End Select
Form6.Hide
End Sub
```

ANEXO 1	Placa del PLC ByL	1



➢ Mechanical diagram



➢ Absolute maximum ratings

Item	Symbol	Min.	Max.	Unit
Supply voltage for logic	Vdd - Vss	0	6.5	V
Input voltage	Vin	0	Vdd	
Operating temperature range	T0p	-20	70	℃
Storage temperature range	Tst	-25	75	

➢ Interface pin connections

Pin No.	Symbol	Level	Description
1	Vdd	5.0V	Supply voltage for logic and LCD (+)
2	Vss	0V	Ground
3	V0	-	Operating voltage for LCD (variable)
4~11	DB0~DB7	H/L	Data bit 0~7
12	CS2	L	Chip select signal for IC2
13	CS1	L	Chip select signal for IC1
14	/RES	L	Reset signal
15	R/W	H/L	H: read (MUP<− module),L: write (MPU− >module)
16	D/I	H/L	H: data, L: instruction code
17	E	H, H L	Chip enable signal
18	VEE	-	Operating voltage for LCD (variable)
19	A	4.2V	Backlight power supply
20	K	0V	Backlight power supply

Electrical Absolute Maximum Ratings (KS0108B)

Parameter	Symbol	Rating	Unit	Note
Operating voltage	V_{DD}	-0.3 ~ +7.0	V	*1
Supply voltage	V_{EE}	V_{DD}-19.0 ~ V_{DD}+0.3	V	*4
Driver supply voltage	V_B	-0.3 ~ V_{DD}+0.3	V	*1,3
	V_{LCD}	V_{EE}-0.3 ~ V_{DD}+0.3	V	*2

***Notes:**

*1. Based on V_{SS} = 0V
*2. Applies the same supply voltage to V_{EE}. V_{LCD}=V_{DD}-V_{EE}
*3. Applies to M, FRM, CLK1, CLK2, CL, RESETB, ADC, CS1B, CS2B, CS3, E, R/W, RS and DB0~DB7.
*4. Applies V0L, V2L, V3L and V5L.

Voltage level: $V_{DD} \geq V0 \geq V1 \geq V2 \geq V3 \geq V4 \geq V5 \geq V_{EE}$

DC Electrical Characteristics (KS0108B)

(VDD= 4.5 to 5.5V, VSS=0V,VDD-VEE=8~17V,Ta= -30 to +85℃)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit	Note
Operating voltage	V_{DD}	-	4.5	-	5.5	V	
Input High voltage	V_{IH1}	-	0.7V_{DD}	-	V_{DD}		*1
	V_{IH2}	-	2.0	-	V_{DD}		*2
Input Low voltage	V_{IL1}	-	0	-	0.3V_{DD}		*1
	V_{IL2}	-	0	-	0.8		*2
Output High Voltage	V_{OH}	I_{OH}= -0.2mA	2.4	-	-		*3
Output Low Voltage	V_{OL}	I_{OL}= 1.6mA	-	-	0.4		*3
Input leakage current	I_{LKG}	V_{IN}= V_{SS} ~ V_{DD}	-1.0	-	+1.0	μA	*4
Three-state (OFF) Input Current	I_{TSL}	V_{IN}= V_{SS} ~ V_{DD}	-5.0	-	5.0		*5
Driver Input leakage current	I_{DIL}	V_{IN}= V_{EE} ~ V_{DD}	-2.0		2.0		*6
On Resistance (Vdiv-Ci)	R_{ONS}	V_{DD}-V_{EE}=15V Load current±100μA	-	-	7.5	kΩ	*8
Operating current	I_{DD1}	During Display	-	-	0.1	mA	*7
	I_{DD2}	During Access Access Cycle=1MHz	-	-	0.5		*7

Notes

*1. CL, FRM, M, RSTB, CLK1, CLK2
*2. CS1B, CS2B, CS3, E, R/W, RS, DB0~DB7
*3. DB0~DB7
*4. Except DB0~DB7
*5. DB0~DB7 at high impedance
*6. V0, V1, V3, V3, V4, V5
*7. 1/64 duty, FCLK=250KHZ, Frame Frequency=70HKZ, Output: No Load
*8. V_{DD}-V_{EE}=15.5V
V0L>V2L>= V_{DD}-2/7(V_{DD}-V_{EE})>V3L= V_{EE}+2/7(V_{DD}-V_{EE})>V5L

DC Electrical Characteristics (KS0107B)

(VDD= 4.5 to 5.5V, VSS=0V,VDD-VEE=8~17V,Ta= -30 to +85℃)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit	Note
Operating voltage	V_{DD}	-	4.5	-	5.5	V	
Input voltage	V_{IH}	-	$0.7V_{DD}$	-	V_{DD}	V	*1
	V_{IL}	-	V_{SS}	-	$0.3V_{DD}$	V	*1
output voltage	V_{OH}	I_{OH}= -0.4mA	V_{DD}-0.4	-	-	V	*2
	V_{OL}	I_{OL}= 0.4mA	-	-	0.4	V	*2
Input leakage current	I_{LKG}	V_{IN}= V_{DD} ~ V_{SS}	-1.0	-	+1.0	μA	*1
OSC Frequency	fosc	Rf=47kΩ±2% Cf=20pF±5%	315	450	585	kHz	
On Resistance (Vdiv-Ci)	R_{ONS}	V_{DD}-V_{EE}=17V Load current±150μA	-	-	1.5	kΩ	
Operating current	I_{DD1}	Master mode 1/128 Duty	-	-	1.0	mA	*3
	I_{DD2}	Master mode 1/128 Duty	-	-	0.2	mA	*4
Supply Current	I_{EE}	Master mode 1/128 Duty	-	-	0.1	mA	*5
Operating	f_{op1}	Master mode External Duty	50	-	600	kHz	
Frequency	f_{op2}	Slave mode	0.5	-	1500	kHz	

Notes

*1. Applies to input terminals FS, DS1, DS2, CR, SHL, MS and PCLK2 and I/O terminals DIO1, DIO2, M, and CL2 in the input state.

*2. Applies to output terminals CLK1, CLK2 and FRM and I/O terminals DIO1, DIO2, M, and CL2 in the output state.

*3. This value is specified about current flowing through V_{SS}.
Internal oscillation circuit: Rf=47kΩ , cf=20pF
Each terminals of DS1, DS2, FS, SHL, and MS is connected to V_{DD} and out is no load.

*4. This value is specified about current flowing through V_{SS}.
Each terminals is DS1, DS2, FS, SHL, PCLK2 and CR is connected to V_{DD},MS is connected to V_{SS} and CL2, M, DIO1 is external clock.

*5. This value is specified about current flowing through V_{EE},Don' t connect to V_{LCD}(V1~V5).

ANEXO 5	Proforma del PLC Unitronics V350	1

DICOMALECTRO CIA. LTDA.
"DISEÑO, CONSTRUCCION Y MANTENIMIENTO CIVIL, ELECTRICO Y ELECTRONICO"

RUC: 1891725457001
DIRECCION: Sucre 641 y Montalvo, Edificio Ortega Oficina 102. Ambato - Ecuador
TELEFONOS: (593) 3-282-4365, 099876021, 092-592-301
CORREO ELECTRONICO: jbalseca@dicomalectro.com
PAGINA WEB: www.dicomalectro.com

OFERTA:	OPLC UNITRONICS V350
COTIZACION Nº:	E061 - 2009
EMPRESA:	
CONTACTO:	Ing. Iván León
DIRECCION:	
TELEF:	
FAX:	
EMAIL:	i_leonguerrero@hotmail.com
FECHA:	12/05/2009

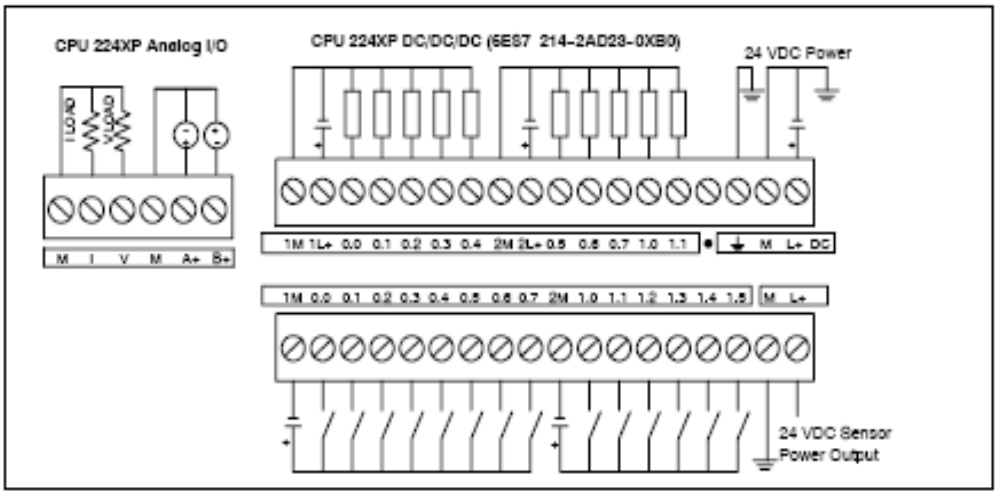
Saludos cordiales, a la vez que le hacemos llegar la siguiente cotización:

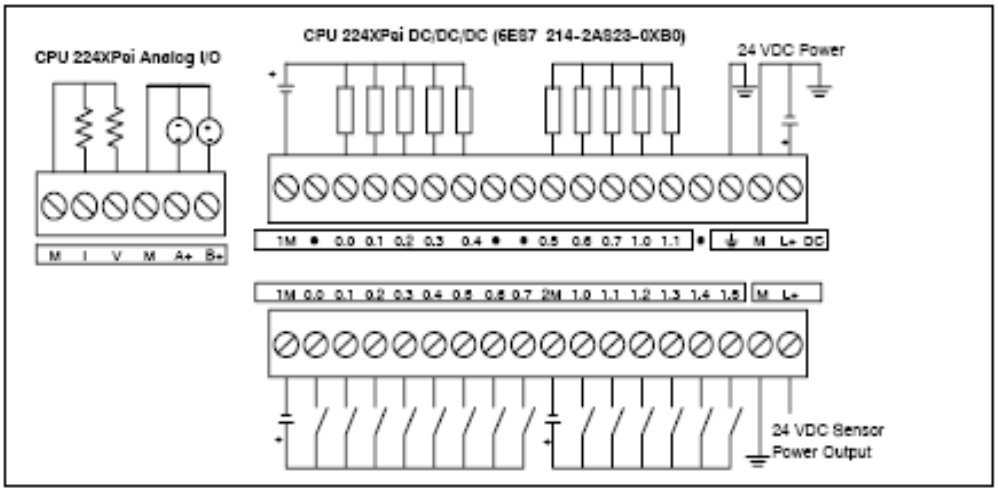
ITEM	CANTIDAD	U/m	DESCRIPCION	MODELO	COSTO U.	COSTO TO
1	1	U	OPLC UNITRONICS VISION V350. Características: Pantalla HMI Touchscreen a color de 3.5", alimentación 24VDC, 8 entradas Digitales, 2 entradas mixtas analógicas/digitales, 2 entradas termocupla/PT100/digitales, 8 salidas relé, 2 salidas Analógicas, 1 puerto de comunicación RS232/RS485 (Modbus) y software de programación gratuito. NOTA: Este OPLC es expandible hasta 512 I/Os locales o remotas.	V350-35-RA22	804,42	804,42
				SUBTOTAL		804,42
				IVA (12%)		96,53
				COSTO FINAL OFERTA (USD)		900,95

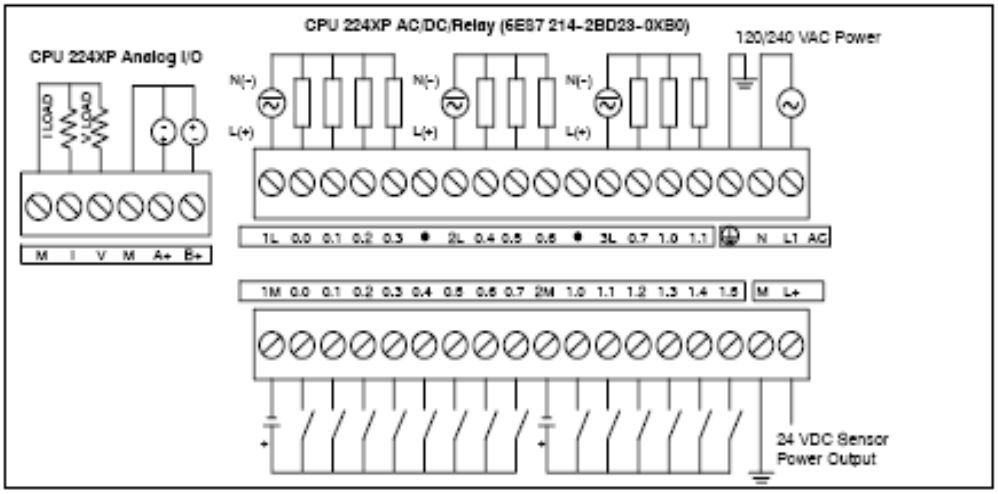
TERMINOS Y CONDICIONES DE LA OFERTA

FORMA DE PAGO:	Adelanto de 60%, entregado el equipo el 40% restante
VALIDEZ OFERTA:	4 semanas contadas a partir de fecha de oferta
LUGAR ENTREGA:	Oficinas Dicomalectro
TIEMPO DE ENTREGA:	2 semanas luego de recepción de orden de compra y entrega de adelanto
OBSERVACIÓN:	Si desea que se incluya la fuente de 24Vcd, sumar $ 55 al subtotal.
REALIZADO POR:	Ing. Fernando Balseca A.

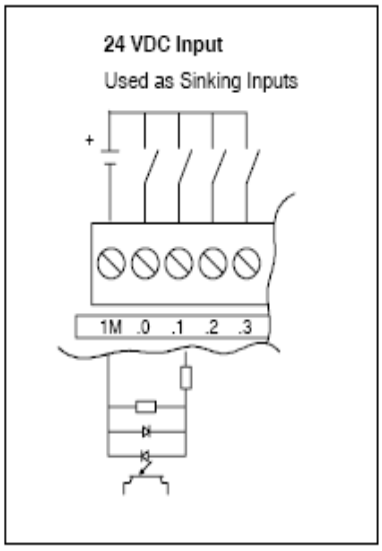
ANEXO 6	Hoja de Características y Proforma del PLC S7-200 CPU 224XP	1

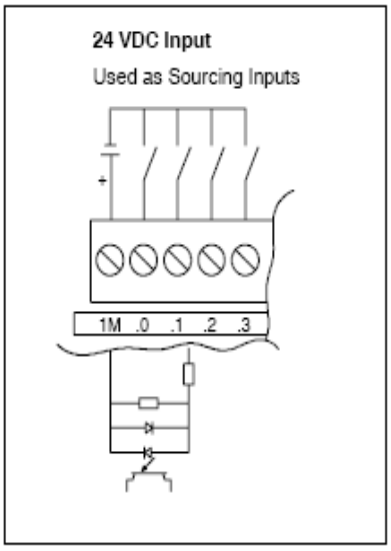


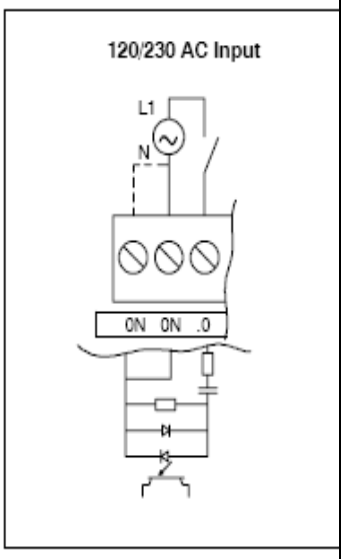




General	24 VDC Input	120/230 VAC Input (47 to 63 HZ)
Type	Sink/Source (IEC Type 1 sink)	IEC Type I
Rated voltage	24 VDC at 4 mA	120 VAC at 6 mA or 230 VAC at 9 mA nominal
Maximum continuous permissible voltage	30 VDC	264 VAC
Surge voltage (max.)	35 VDC for 0.5 s	-
Logic 1 (min.)	15 VDC at 2.5 mA	79 VAC at 2.5 mA
Logic 0 (max.)	5 VDC at 1 mA	20 VAC or 1 mA AC
Input delay (max.)	4.5 ms	15 ms
Connection of 2 wire proximity sensor (Bero) Permissible leakage current (max.)	 1 mA	 1 mA AC
Isolation Optical (galvanic, field to logic) Isolation groups	 500 VAC for 1 minute See wiring diagram	 1500 VAC for 1 minute 1 point
Inputs on simultaneously	All at 55° C (horizontal), All on at 45° C (vertical)	
Cable length (max.) Shielded Unshielded	 500 m 300 m	 500 m 300 m







General	24 VDC Output		Relay Output		120/230 VAC Outp
	0.75 A	5 A	2 A	10 A	
Type	Solid state-MOSFET (Sourcing)		Dry contact		Triac, zero-cross turn
Rated voltage	24 VDC		24 VDC or 250 VAC		120/230 VAC
Voltage range	20.4 to 28.8 VDC		5 to 30 VDC or 5 to 250 VAC	12 to 30 VDC or 12 to 250 VAC	40 to 264 VAC (47 to 63 Hz)
24 VDC coil power voltage range	-		20.4 to 28.8 VDC		-
Surge current (max.)	8 A for 100 ms	30 A	5 A for 4 s @ 10% duty cycle	15 A for 4 s @ 10% duty cycle	5 A rms for 2 AC cycl
Logic 1 (min.)	20 VDC		-		L1 (-0.9 V rms)
Logic 0 (max.)	0.1 VDC with 10 K Ω Load	0.2 VDC with 5 K Ω Load	-		-
Rated current per point (max.)	0.75 A	5 A	2.00 A	10 A resistive; 2 A DC inductive; 3 A AC inductive	0.5 A AC[1]
Rated current per common (max.)	10 A	5 A	10 A	10 A	0.5 A AC
Leakage current (max.)	10 µA	30 µA	-		1.1 mA rms at 132 VA and 1.8 mA rms at 2 VAC
Lamp load (max.)	5 W	50 W	30 W DC/ 200 W AC[4,5]	100 W DC/ 1000 W AC	60 W
Inductive clamp voltage	L+ minus 48 V	L+ minus 47 V[2]	-		-
On state resistance (contact)	0.3 Ω typical (0.6 Ω max.)	0.05 Ω max.	0.2 Ω max. when new	0.1 Ω max. when new	410 Ω max. when loa current is less than 0.05A
Isolation					
Optical (galvanic, field to logic)	500 VAC for 1 minute		-		1500 VAC for 1minute
Coil to logic	-		None		-
Coil to contact	-		1500 VAC for 1 minute		-
Resistance (coil to contact)	-		100 M Ω min. when new		-
Isolation groups	See wiring diagram		See wiring diagram		1 point
Delay Off to On/On to Off (max.)	50 µs / 200 µs	500 µs	-	-	0.2 ms + 1/2 AC cycle
Switching (max.)	-	-	10 ms	15 ms	-
Switching frequency (max.)	-		1 Hz		10 Hz
Lifetime mechanical cycles	-		10,000,000 (no load)	30,000,000 (no load)	-
Lifetime contacts	-		100,000 (rated load)	30,000 (rated load)	-
Output on simultaneously	All at 55° C (horizontal), All at 45° C (vertical)			All at 55 °C (horizontal) with 20A max. module current. All at 45°C (vertical) with 20A max. module current[3]. All at 40 °C (horizontal) with 10A per point	All at 55° C (horizonta All at 45° C (vertical)
Connecting two outputs in parallel	Yes, only outputs in same group		No		No
Cable length (max.)					
Shielded	500 m		500 m		500 m
Unshielded	150 m		150 m		150 m

1 Load current must be full wave AC and must not be half-wave because of the zero-cross circuitry. Minimum load current is 0.05 A AC. With a load cu between 5 mA and 50 mA AC, the current can be controlled, but there is an additional voltage drop due to series resistance of 410 Ohms.

2 If the output overheats due to excessive inductive switching or abnormal conditions, the output point may turn off or be damaged. The output could over or be damaged if the output is subjected to more than 0.7 J of energy switching an inductive load off. To eliminate the need for this limitation, a suppres circuit as described in Chapter 3 can be added in parallel with the load. These components need to be sized properly for the given application.

3 The EM 222 DO 4 x Relay has a different FM rating than the rest of the S7-200. This module has a T4 rating, instead of T4A for FM Class I, Division Gr A, B, C, and D Hazardous Locations.

4 Relay lifetime with a lamp load will be reduced by 75% unless steps are taken to reduce the turn-on surge below the surge current rating of the outpu

5 Lamp load wattage rating is for rated voltage. Reduce the wattage rating proportionally for voltage being switched (for example 120 VAC - 100 W).

ANEXO 7	Diagramas Esquemáticos de las Tarjetas del PLC	1

Se presenta a continuación.

MANUAL PLC ByL