ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN TECNOLÓGICA

PROTOTIPO EDUCATIVO PARA EL LABORATORIO DE MICROPROCESADORES USANDO UN MICROCONTROLADOR PIC Y EL MAX 232.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO DE ELECTRÓNICA Y TELECOMUNICACIONES

GUAMÁN CONDE DIANA FERNANDA

didi281285@hotmail.com

OÑA CHIPUXI ROSA YADIRA

yadyross@hotmail.com

DIRECTOR: ING. ALCÍVAR COSTALES

eduardo.costales@epn.edu.ec

Quito, Marzo del 2011

DECLARACIÓN

Nosotras, Diana Fernanda Guamán Conde y Rosa Yadira Oña Chipuxi, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Diana Fernanda Guamán C.

Rosa Yadira Oña Ch.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Diana Fernanda Guamán Conde y Rosa Yadira Oña Chipuxi, bajo mi supervisión.

Ing. Alcivar Costales.
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Primero agradezco a Dios, por ser mi guía en cada una de las decisiones que he tomado en mi vida.

Agradezco a mi madre Rosa Amelia Conde, por ser siempre mi apoyo incondicional, y ser mi más grande fuente de inspiración para seguir preparándome, y por ella he podido culminar esta etapa de mi vida. A mi padre Manuel Raúl, que a pesar de la distancia siempre estuvo atento para saber cómo iba mi proceso.

Alex, lo único que te puedo decir, es que de todo corazón te agradezco el tenerme toda la paciencia, confiar en mí y estar siempre a mi lado.

Gracias a todos mis amigos que en el momento más difícil de mi vida supieron brindarme su apoyo y comprensión para seguir adelante, también a todas aquellas personas que no están ahora conmigo pero que en su momento indirectamente participaron de esta gran aventura.

Este presente proyecto, es el resultado del esfuerzo y dedicación de parte de la autora y su director de tesis, sin su guía no hubiese sido posible la culminación del mismo.

Diana Guamán Conde

AGRADECIMIENTOS

Agradezco a Dios por darme salud, sabiduría y fuerza para culminar este presente proyecto.

A mis padres, ya que me han brindado su apoyo y comprensión incondicional en las buenas y en las malas, y gracias a ellos pude culminar esta etapa de mi vida.

A mis maestros y al Ing. Alcívar Costales que han impartido sus conocimientos hacia mi persona, gracias a ellos he adquirido conocimientos para culminar este proyecto.

Rosa Yadira Oña Ch.

DEDICATORIA

Este proyecto lo dedico con todo mi amor a mis padres que a pesar de los momentos difíciles que hemos pasado ellos siguen confiando en mí y brindándome el soporte para seguir adelante. Gracias mamá por pensar siempre en mi futuro y apoyarme en cada nueva meta que me propongo.

A mi tío Leyo Conde que a pesar de la distancia me ha dado su apoyo total y la fuerza moral para seguir adelante.

Diana Guamán Conde

DEDICATORIA

Este presente proyecto dedico:

Principalmente a mis padres José Miguel Oña y Rosa María Chipuxi, ya que ellos han sido mi guía y el pilar para alcanzar mis metas.

A mi hija Nayerly ya que ella me ha dado las fuerzas suficientes para seguir adelante.

Rosa Yadira Oña Ch.

INDICE

INTRODUCCIÓN

El presente proyecto de titulación se basa en la implementación de un prototipo educativo de control computarizado en una maqueta para la automatización de una casa, mediante un microcontrolador PIC 16F877A, con interfaz serial RS – 232, este lleva un control desde la PC hacia diferentes áreas de la casa por medio del Visual Basic que será el interfaz entre el PIC y el usuario, Visual Basic se comunicará con el PIC por medio del puerto serial RS-232 para que se haga efectiva la comunicación vamos a usar el MAX 232.

El tener el control desde la PC nos ahorra tiempo porque desde un mismo sitio podemos administrar diferentes áreas de nuestra casa, en este caso vamos a controlar las luces a que se prendan y se apaguen a una determinada hora, vamos a implementar una alarma y un sensor de humo para la seguridad de la misma, si en el caso se activara la alarma nosotros vamos a observar en que área fue activada ya que sabremos el lugar preciso que hizo que la alarma se activara.

Con el presente proyecto se trata de que los mismos estudiantes puedan programar su PIC y puedan ver los resultados por medio de la maqueta, y así de esta manera fortalecer sus conocimientos en la materia de Control con Microprocesadores.

El presente proyecto se dividió en cinco capítulos: En el primer capítulo nos basaremos en el hardware, en el segundo capítulo en el software, el tercer capítulo en el diseño y ensamblaje del proyecto, en el cuarto capítulo es de pruebas y placas y por último, el quinto capítulo es de conclusiones y recomendaciones.

CAPÍTULO I:
FUNDAMENTOS TEÓRICOS.

1.1. RESISTENCIA ELÉCTRICA[1]

Resistencia eléctrica es toda oposición que encuentra la corriente a su paso por un circuito eléctrico cerrado, atenuando o frenando el libre flujo de circulación de las cargas eléctricas o electrones. Cualquier dispositivo o consumidor conectado a un circuito eléctrico representa en sí una carga, resistencia u obstáculo para la circulación de la corriente eléctrica.

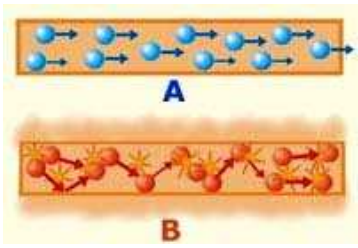


Figura 1.1: Circulación de la corriente sobre la resistencia.

A.- Electrones fluyendo por un buen conductor eléctrico, que ofrece baja resistencia
B.- Electrones fluyendo por un mal conductor eléctrico, que ofrece alta resistencia a su paso. En ese caso los electrones chocan unos contra otros al no poder circular libremente y, como consecuencia, generan calor.

Se dividen en tres grupos:

- **Resistencias lineales fijas:** Su valor de resistencia es constante y esta predeterminado por el fabricante.
- **Resistencias variables:** Su valor de resistencia puede variar dentro de unos límites.
- **Resistencia no lineal:** Su valor de resistencia varía de forma no lineal dependiendo de distintas magnitudes físicas (temperatura, luminosidad, etc.)

[1] http://www.asifunciona.com/electrotecnia/ke_resistencia/ke_resistencia_1.htm

1.2. CONDENSADORES[2]

Básicamente un condensador es un dispositivo capaz de almacenar energía en forma de campo eléctrico. Está formado por dos armaduras metálicas paralelas (generalmente de aluminio) separadas por un material dieléctrico.
Va a tener una serie de características tales como capacidad, tensión de trabajo, tolerancia y polaridad, que deberemos aprender a distinguir.

Un condensador con las dos láminas, placas o armaduras, y el dieléctrico entre ellas. En la versión más sencilla del condensador, no se pone nada entre las armaduras y se las deja con una cierta separación, en cuyo caso se dice que el dieléctrico es el aire.

- **Tensión de trabajo**: Es la máxima tensión que puede aguantar un condensador, que depende del tipo y grosor del dieléctrico con que esté fabricado. Si se supera dicha tensión, el condensador puede perforarse (quedar cortocircuitado) y/o explotar. En este sentido hay que tener cuidado al elegir un condensador, de forma que nunca trabaje a una tensión superior a la máxima.
- **Tolerancia**: Igual que en las resistencias, se refiere al error máximo que puede existir entre la capacidad real del condensador y la capacidad indicada sobre su cuerpo.
- **Polaridad**: Los condensadores electrolíticos y en general los de capacidad superior a 1 µF tienen polaridad, eso es, que se les debe aplicar la tensión prestando atención a sus terminales positivo y negativo. Al contrario que los inferiores a 1µF, a los que se puede aplicar tensión en cualquier sentido, los que tienen polaridad pueden explotar en caso de ser ésta la incorrecta.

[2] http://www.planetaelectronico.com/cursillo/tema2/tema2.3.html

- **Capacidad**: Se mide en Faradios (**F**), aunque esta unidad resulta tan grande como los submúltiplos, tales como microfaradios (**μF**=10^{-6} F), nanofaradios (**nF**=10^{-9} F) y picofaradios (**pF**=10^{-12} F).

1.3. TRANSISTOR[3]

Dispositivo compuesto de un material semiconductor que amplifica una señal o abre o cierra un circuito. Inventado en 1947 en Bell Labs, los transistores se han vuelto el principal componente de todos los circuitos digitales, incluidas las computadoras. En la actualidad los microprocesadores contienen millones de transistores microscópicos.

Previo a la invención de los transistores, los circuitos digitales estaban compuestos de tubos vacíos, lo cual tenía muchas desventajas. Eran más grandes, requerían más memoria y energía, generaban más calor y eran más propensos a fallas.

Los transistores cumplen las funciones de amplificador, oscilador, conmutador o rectificador. El transistor bipolar fue inventado en los laboratorios Bell de EEUU en diciembre de 1947 por John Bardeen, Walter Houser Brattain y William Bradford Shockley (recibieron el Premio Nobel de Física en 1956).

1.3.1. REGIONES OPERATIVAS DEL TRANSISTOR[3]

Los transistores bipolares de juntura tienen diferentes regiones operativas, definidas principalmente por la forma en que son polarizados:

1.3.1.1. Región activa[3]

Cuando un transistor no está ni en su región de saturación ni en la región de corte entonces está en una región intermedia, la región activa. En esta región la

[3] http://es.wikipedia.org/wiki/Transistor_de_uni%C3%B3n_bipolar

corriente de colector (Ic) depende principalmente de la corriente de base (Ib), de β (ganancia de corriente, es un dato del fabricante) y de las resistencias que se encuentren conectadas en el colector y emisor. Esta región es la más importante si lo que se desea es utilizar el transistor como un amplificador de señal.

1.3.1.2. Región inversa[3]

Al invertir las condiciones de polaridad del funcionamiento en modo activo, el transistor bipolar entra en funcionamiento en modo inverso. En este modo, las regiones del colector y emisor intercambian roles. Debido a que la mayoría de los TBJ son diseñados para maximizar la ganancia de corriente en modo activo, el parámetro beta en modo inverso es drásticamente menor al presente en modo activo.

1.3.1.3. Región de corte[3]

Un transistor esta en corte cuando:
Corriente de colector = corriente de emisor = 0, (Ic = *Ie* = 0). En este caso el voltaje entre el colector y el emisor del transistor es el voltaje de alimentación del circuito. (Como no hay corriente circulando, no hay caída de voltaje, ver Ley de Ohm). Este caso normalmente se presenta cuando la corriente de base = 0 (Ib =0)

1.3.1.4. Región de saturación[3]

Un transistor está saturado cuando:
Corriente de colector = corriente de emisor = corriente máxima, (Ic = Ie = Imaxima). En este caso la magnitud de la corriente depende del voltaje de alimentación del circuito y de las resistencias conectadas en el colector o el emisor o en ambos, ver ley de Ohm. Este caso normalmente se presenta cuando la corriente de base es lo suficientemente grande como para inducir una corriente de colector β veces más grande. (recordar que Ic =β * Ib)

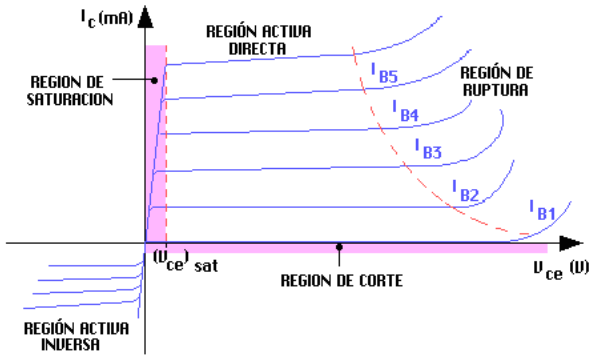


Figura 1.2: Regiones de operación del BJT.

1.4. OPTOACOPLADORES (OPTOAISLADORES)[4]

Un optó acoplador es un componente formado por la unión de al menos un emisor (diodo LED) y un fotodetector (fototransistor u otro) acoplados a través de un medio conductor de luz, pueden ser encapsulados o de tipo discreto.

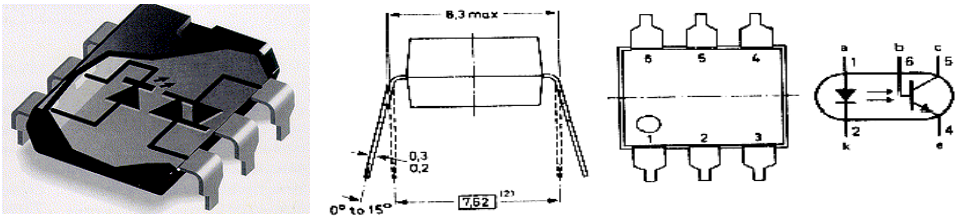


Figura 1.3: Estructura externa e interna de un optoacoplador.

1.4.1. ESQUEMA DE UN OPTOACOPLADOR

Cuanta mayor intensidad atraviesa el fotodiodo, mayor será la cantidad de fotones emitidos y, por tanto, mayor será la corriente que recorra el fototransistor. Se trata de una manera de transmitir una señal de un circuito eléctrico a otro. Obsérvese que no existe comunicación eléctrica entre los dos circuitos, es decir existe un trasiego de información pero no existe una conexión eléctrica: la conexión es óptica.

[4] ttp://www.elprisma.com/apuntes/ingenieria_electrica_y_electronica/electronicadepotencia/default5.asp

Las implementaciones de un optó acoplador son variadas y dependen de la casa que los fabrique. Una de las más populares se ve en la Figura 1.4. Se puede observar como el LED, en la parte superior, emite fotones que, tras atravesar el vidrio, inciden sobre el fototransistor.

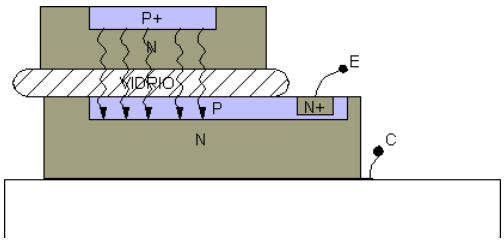


Figura 1.4: Esquema constructivo de un optoacoplador.

1.4.2. ESTRUCTURA [4]

La Figura 1.5 muestra la perspectiva interna de un optó acoplador. Una resina aloja al elemento sensitivo a la luz (fototransistor o fototransistor de salida Darlington) que está rodeado por otra resina que permite la transmisión de la luz. Una señal luminosa es transmitida por un diodo emisor de luz hacia el transistor fotosensitivo a través de la resina transmisora de luz interna.

La resina albergue y la resina interior tienen el mismo coeficiente de expansión. El alto aislamiento de voltaje se obtiene gracias al gran área existente entre la resina externa y la interna que no es modificada por los cambios de temperatura pues los coeficientes de expansión son iguales, además, si la temperatura aumenta las resinas se expanden obteniéndose como resultado una mayor área entre los elementos conductores.

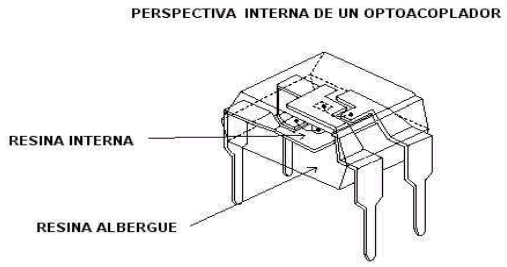


Figura 1.5: Estructura interna de un optoacoplador.

1.4.3. DIFERENTES TIPOS DE OPTOACOPLADORES[4]

1.4.3.1. Fototransistor[4]

Se compone de un optoacoplador con una etapa de salida formada por un transistor BJT. Se trata de un transistor bipolar sensible a la luz.

Figura 1.6: Símbolo del fototransistor.

La radiación luminosa se hace incidir sobre la unión colector base cuando éste opera en la RAN. En esta unión se generan los pares electrón - hueco, que provocan la corriente eléctrica.

El funcionamiento de un fototransistor viene caracterizado por los siguientes puntos:

- Un fototransistor opera, generalmente sin terminal de base (I_b=0) aunque en algunos casos hay fototransistores tienen disponible un terminal de base para trabajar como un transistor normal.
- La sensibilidad de un fototransistor es superior a la de un fotodiodo, ya que la pequeña corriente foto generada es multiplicada por la ganancia del transistor.
- Las curvas de funcionamiento de un fototransistor son las que aparecen en la Figura 1.7. Como se puede apreciar, son curvas análogas a las del transistor BJT, sustituyendo la intensidad de base por la potencia luminosa por unidad de área que incide en el fototransistor.

Figura 1.7: Curvas características de un fototransistor típico.

1.4.3.2. Fototriac[4]

Se compone de un optó acoplador con una etapa de salida formada por un triac.

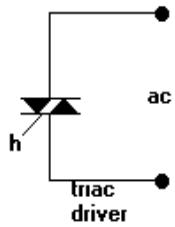


Figura 1.8: Símbolo del fototriac.

1.4.3.3. Fototriac de paso por cero[4]

Optó acoplador en cuya etapa de salida se encuentra un triac de cruce por cero. El circuito interno de cruce por cero conmuta al triac sólo en los cruce por cero de la corriente alterna.

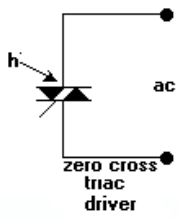


Figura 1.9: Símbolo del fototriac de paso por cero.

1.4.4. FUNCIONAMIENTO[4]

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. Estos son muy útiles cuando se utilizan por ejemplo, Microcontroladores PICs y/o PICAXE si queremos proteger nuestro microcontrolador este dispositivo es una buena opción. En general pueden sustituir los relés ya que tienen una velocidad de conmutación mayor, así como, la ausencia de rebotes. 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 que por lo general es del tipo DIP.

1.4.5. TIPOS MÁS COMUNES

4N26 - 4N33 –MOC3020- MOC3021 - MOC3041 - MOC3163 - ECG3048.

1.5. EL TRIAC[5]

1.5.1. INTRODUCCIÓN[5]

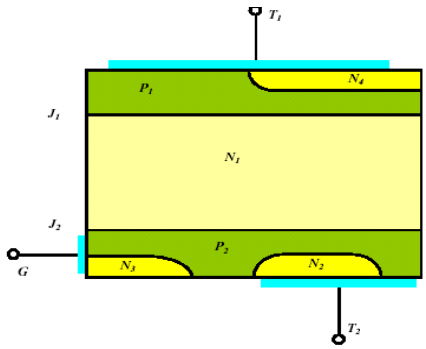
El triac es un dispositivo semiconductor de tres terminales que se usa para controlar el flujo de corriente promedio a una carga, con la particularidad de que conduce en ambos sentidos y puede ser bloqueado por inversión de la tensión o al disminuir la corriente por debajo del valor de mantenimiento. El triac puede ser disparado independientemente de la polarización de puerta, es decir, mediante una corriente de puerta positiva o negativa.

[5] http://www.monografias.com/trabajos14/triac/triac.shtml

1.5.2. DESCRIPCIÓN GENERAL[5]

Cuando el triac conduce, hay una trayectoria de flujo de corriente de muy baja resistencia de una terminal a la otra, dependiendo la dirección de flujo de la polaridad del voltaje externo aplicado. Cuando el voltaje es más positivo en MT2, la corriente fluye de MT2 a MT1 en caso contrario fluye de MT1 a MT2. En ambos casos el triac se comporta como un interruptor cerrado. Cuando el triac deja de conducir no puede fluir corriente entre las terminales principales sin importar la polaridad del voltaje externo aplicado por tanto actúa como un interruptor abierto.
Debe tenerse en cuenta que si se aplica una variación de tensión importante al triac (dv/dt) aún sin conducción previa, el triac puede entrar en conducción directa.

1.5.3. CONSTRUCCIÓN BÁSICA, SÍMBOLO, DIAGRAMA EQUIVALENTE[5]



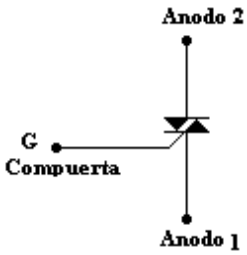


Figura 1.10: Construcción interna del triac. **Figura 1.11:** Símbolo del triac.

La estructura contiene seis capas, aunque funciona siempre como un tiristor de cuatro capas. En sentido MT2-MT1 conduce a través de P1N1P2N2 y en sentido MT1-MT2 a través de P2N1P1N4. La capa N3 facilita el disparo con intensidad de puerta negativa. La complicación de su estructura lo hace más delicado que un tiristor en cuanto a di/dt y dv/dt y capacidad para soportar sobre intensidades. Se fabrican para intensidades de algunos amperios hasta unos 200 A eficaces y desde 400 a 1000 V de tensión de pico repetitivo. Los triac son fabricados para funcionar a frecuencias bajas, los fabricados para trabajar a frecuencias medias son denominados alternistores. En la Figura 1.11 se muestra el símbolo esquemático e identificación de las terminales de un triac, la nomenclatura ánodo

2 (A2) y ánodo 1 (A1) pueden ser reemplazados por Terminal Principal 2 (MT2) y Terminal Principal 1 (MT1) respectivamente.

El Triac actúa como dos rectificadores controlados de silicio (SCR) en paralelo Figura 1.12, este dispositivo es equivalente a dos latchs.

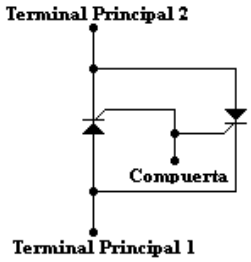


Figura 1.12: Triac equivalente a dos latch.

1.5.4. CARACTERÍSTICA TENSIÓN – CORRIENTE[5]

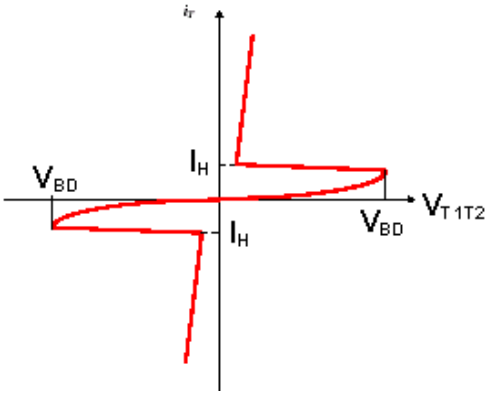


Figura 1.13: Curva característica tensión – corriente del triac.

La Figura 1.13 describe la característica tensión – corriente del Triac. Muestra la corriente a través del Triac como una función de la tensión entre los ánodos MT2 y MT1.

El punto VBD (tensión de ruptura) es el punto por el cual el dispositivo pasa de una resistencia alta a una resistencia baja y la corriente, a través del Triac, crece con un pequeño cambio en la tensión entre los ánodos.

El Triac permanece en estado ON hasta que la corriente disminuye por debajo de la corriente de mantenimiento IH. Esto se realiza por medio de la disminución de la tensión de la fuente. Una vez que el Triac entra en conducción, la compuerta no controla más la conducción, por esta razón se acostumbra dar un pulso de corriente corto y de esta manera se impide la disipación de energía sobrante en la compuerta.

El mismo proceso ocurre con respecto al tercer cuadrante, cuando la tensión en el ánodo MT2 es negativa con respecto al ánodo MT1 y obtenemos la característica invertida. Por esto es un componente simétrico en cuanto a conducción y estado de bloqueo se refiere, pues la característica en el cuadrante I de la curva es igual a la del III.

1.6. EL MICROCONTROLADOR PIC

1.6.1. EL MICROCONTROLADOR[6]

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un computador, esto es CPU, memoria RAM, EEPROM y circuitos de entrada y salida.

Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado para que realice desde un simple parpadeo de un led hasta un sofisticado control de robot. Un microcontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertas AND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc., simplificando todo el diseño a una placa de reducido tamaño y pocos elementos.

[6] REYES, Carlos.A.Microcontroladores Pic, Programacion en Basic, Segunda Edición, Editora Rispergraf, pag: 17

1.6.2. DIFERENCIA ENTRE MICROPROCESADOR Y MICROCONTROLADOR[7]

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de un computador. La UCP está formada por la Unidad de Control, que interpreta las instrucciones, y el Camino de Datos, que las ejecuta.

Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine.

Un microprocesador es un sistema abierto con el que puede construirse un computador con las características que se desee, acoplándole los módulos necesarios. Un microcontrolador es un sistema cerrado que contiene un computador completo y de prestaciones limitadas que no se pueden modificar.

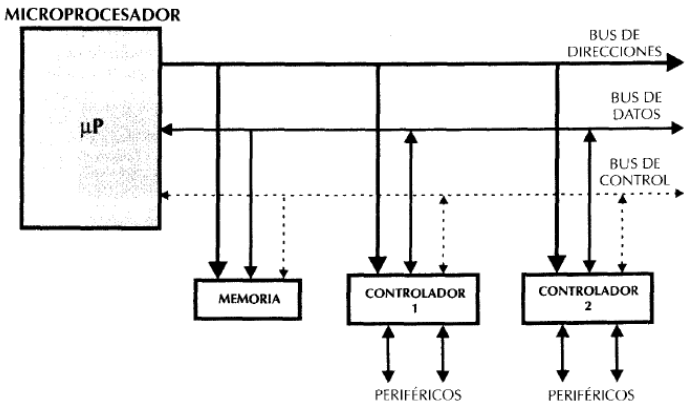


Figura 1.14: Estructura de un sistema abierto basado en un microprocesador.

La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.

[7] ANGULO Jose,Angulo Ignacio. Microcontroladores PIc, Diseño Practico de Aplicaciones, Tercera edición, Editorial McGrawHill, Pag. 3-9.

.

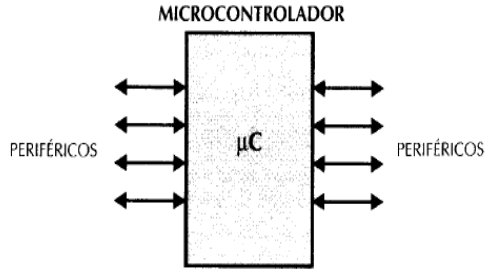


Figura 1.15: El microcontrolador en un sistema cerrado.

Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.
Si sólo se dispusiese de un modelo de microcontrolador, éste debería tener muy potenciados todos sus 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 cantidad y potencia de los elementos auxilares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

1.6.3. ARQUITECTURA INTERNA[7]

Un microcontrolador posee todos los componentes de un computador, pero con unas características fijas que no pueden alterarse.

Las partes principales de un microcontrolador son:

1. Procesador
2. Memoria no volátil para contener el programa
3. Memoria de lectura y escritura para guardar los datos
4. Líneas de EIS para los controladores de periféricos:

a) Comunicación paralelo

b) Comunicación serie

c) Diversas puertas de comunicación (bus I2ºC, USB, etc.)

5. Recursos auxiliares:

a) Circuito de reloj

b) Temporizadores

c) Perro Guardián («watchdog»)

d) Conversores AD y DA

e) Comparadores analógicos

f) Protección ante fallos de la alimentación

g) Estado de reposo o de bajo consumo.

A continuación se revisará las características más representativas de cada uno de los componentes del microcontrolador.

1.6.3.1. El procesador[7]

La necesidad de conseguir elevados rendimientos en el procesamiento de las instrucciones ha desembocado en el empleo generalizado de procesadores de arquitectura Harvard frente a los tradicionales que seguían la arquitectura de Von Neumann. Esta última se caracterizaba porque la UCP (Unidad Central de Proceso) se conectaba con una memoria única, donde coexistían datos e instrucciones, a través de un sistema de buses.

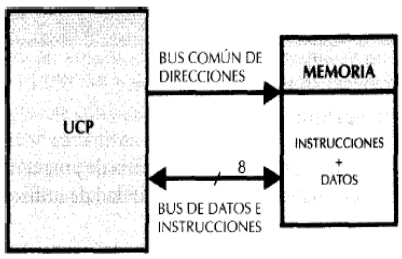


Figura 1.16: En la arquitectura de «Von Neumann» la UCP se comunicaba a través de un sistema de buses con la Memoria, donde se guardaban las instrucciones y los datos.

En la arquitectura Harvard son independientes la memoria de instrucciones y la memoria de datos y cada una dispone de su propio sistema de buses para el acceso. Esta dualidad, además de propiciar el paralelismo, permite la adecuación del tamaño de las palabras y los buses a los requerimientos específicos de las instrucciones y de los datos. También la capacidad de cada memoria es diferente.

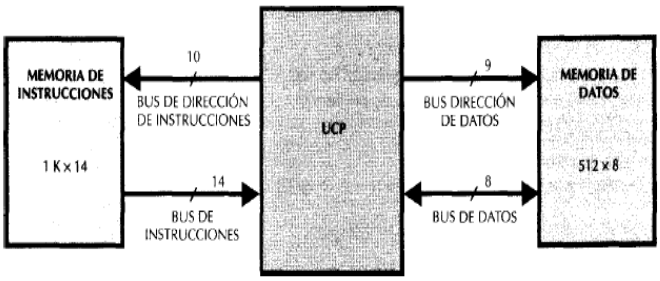


Figura 1.17: Arquitectura Harward.

En la Figura1.17, la memoria de instrucciones y la de datos son independientes, lo que permite optimizar sus características y propiciar el paralelismo, además la memoria de instrucciones tiene 1K posiciones de 14 bits cada una, mientras que la de datos solo dispone de 14 bits cada una, mientras que la de datos solo dispone de 512 posiciones de un byte.

El procesador de los modernos microcontroladores responde a la arquitectura RISC (Computadores de Juego de Instrucciones Reducido), que se identifica por poseer un repertorio de instrucciones máquina pequeño y simple, de forma que la mayor parte de las instrucciones se ejecuta en un ciclo de instrucción. Otra aportación frecuente que aumenta el rendimiento del computador es el fomento del paralelismo implícito, que consiste en la segmentación del procesador (pipe-line*),* descomponiéndolo en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.

El alto rendimiento y elevada velocidad que alcanzan los modernos procesadores, como el que poseen los microcontroladores PIC, se debe a la conjunción de tres técnicas:

- Arquitectura Harvard
- Computador tipo RISC
- Segmentación

1.6.3.2. Memoria de programa[7]

El microcontrolador está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. No hay posibilidad de utilizar memorias externas de ampliación.
Como el programa a ejecutar siempre es el mismo, debe estar grabado de forma permanente.
Los tipos de memoria adecuados para soportar esta función admiten cinco versiones diferentes:

- **ROM con máscara**

En este tipo de memoria el programa se graba en el chip durante el proceso de su fabricación mediante el uso de «máscaras». Los altos costes de diseño e instrumental solo aconsejan usar este tipo de memoria cuando se precisan series muy grandes.

- **EPROM**

La grabación de esta memoria se realiza mediante un dispositivo físico gobernado desde un computador personal, que recibe el nombre de grabador. En la superficie de la cápsula del microcontrolador existe una ventana de cristal por la que se puede someter al chip de la memoria a rayos ultravioletas para producir su borrado y emplearla nuevamente. Es interesante la memoria EPROM en la fase de diseño y depuración de los programas, pero su coste unitario es elevado.

- **OTP (Programable una vez)**

Este modelo de memoria sólo se puede grabar una vez por parte del usuario, utilizando el mismo procedimiento que con la memoria EPROM. Posteriormente no se puede borrar. Su bajo precio y la sencillez de la grabación aconsejan este tipo de memoria para prototipos finales y series de producción cortas.

- **EEPROM**

La grabación es similar a las memorias OTP y EPROM, pero el borrado es mucho más sencillo al poderse efectuar de la misma forma que el grabado, o sea, eléctricamente. Sobre el mismo zócalo del grabador puede ser programada y borrada tantas veces como se quiera, lo cual la hace ideal en la enseñanza y en la creación de nuevos proyectos.

- **FLASH**

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar en circuito al igual que las EEPROM, pero suelen disponer de mayor capacidad que estas últimas.El borrado sólo es posible con bloques completos y no se puede realizar sobre posiciones concretas. En las FLASH se garantizan 1.000 ciclos de escritura-borrado.

Son muy recomendables en aplicaciones en las que sea necesario modificar el programa a lo largo de la vida del producto, como consecuencia del desgaste o cambios de piezas, como sucede con los vehículos.

Por sus mejores prestaciones está sustituyendo a la memoria EEPROM para contener instrucciones. De esta forma Microchip comercializa dos microcontroladores prácticamente iguales, que sólo se diferencian en que la memoria de programa de uno de ellos es tipo eeprom y la del otro tipo flash. se trata del PIC 16c84 y el PIC 16f84, respectivamente. En la actualidad microchip tiene abierta una línea de PIC con memoria flash cada vez más extensa y utilizada.

1.6.3.3. Memoria de datos[7]

Los datos que manejan los programas varían continuamente, y esto exige que la memoria que les contiene debe ser de lectura y escritura, por lo que la memoria RAM estática (SRAM) es la más adecuada, aunque sea volátil.
Hay microcontroladores que también disponen como memoria de datos una de lectura y escritura no volátil, del tipo EEPROM. De esta forma, un corte en el

suministro de la alimentación no ocasiona la pérdida de la información, que está disponible al reiniciarse el programa.

1.6.3.4. Líneas de E/S para los controladores de periféricos[7]

A excepción de dos pines destinados a recibir la alimentación, otras dos para el cristal de cuarzo, que regula la frecuencia de trabajo, y una más para provocar el Reset, los restantes pines de un microcontrolador sirven para soportar su comunicación con los periféricos externos que controla.

Las líneas de E/S que se adaptan con los periféricos manejan información en paralelo y se agrupan en conjuntos de ocho, que reciben el nombre de Puertas. Hay modelos con líneas que soportan la comunicación en serie; otros disponen de conjuntos de líneas que implementan puertas de comunicación para diversos protocolos, como el I2ºC, el USB, etc.

1.6.3.5. Recursos auxiliares[7]

Según las aplicaciones a las que orienta el fabricante cada modelo de microcontrolador, incorpora una diversidad de complementos que refuerzan la potencia y la flexibilidad del dispositivo. Entre los recursos más comunes se citan a los siguientes:

a) Circuito de reloj, encargado de generar los impulsos que sincronizan el funcionamiento de todo el sistema.
b) Temporizadores, orientados a controlar tiempos.
c) Perro Guardián («watchdog»), destinado a provocar una reinicialización cuando el programa queda bloqueado.
d) Conversores AD y DA, para poder recibir y enviar señales analógicas.
e) Comparadores analógicos, para verificar el valor de una señal analógica.
f) Sistema de protección ante fallos de la alimentación.
g) Estado de Reposo, en el que el sistema queda «congelado» y el consumo de energía se reduce al mínimo.

1.7. COMUNICACIÓN SERIAL[8]

El puerto serial de las computadoras es conocido como puerto RS-232, la ventaja de este puerto es que todas las computadoras traen al menos un puerto serial, este permite las comunicaciones entre otros dispositivos tales como otra computadora, el mouse, impresora y para nuestro caso con los microcontroladores.

Existen dos formas de intercambiar información binaria: la paralela y la serial.

La comunicación paralela transmite todos los bits de un dato de manera simultánea, por lo tanto la velocidad de transferencia es rápida, sin embargo tiene la desventaja de utilizar una gran cantidad de líneas, por lo tanto se vuelve más costoso y tiene la desventaja de atenuarse a grandes distancias, por la capacitancia entre conductores así como sus parámetros distribuidos.

1.7.1. LÍNEAS O CANALES DE COMUNICACIÓN[9]

Se pueden establecer canales para la comunicación de acuerdo a tres técnicas, siempre tomando al microprocesador o microcontrolador como referencia (transmisor) y al periférico como destino (receptor):

a. Simplex
b. Semi duplex (Half duplex)
c. Totalmente duplex (Full duplex)

[8] http://www.i-micro.com/pdf/articulos/rs-232.pdf

[9] http://perso.wanadoo.es/pictob/comserie.htm

1.7.1.1. Simplex[9]

En ella la comunicación serie usa una dirección y una línea de comunicación. Siempre existirá un transmisor y un receptor, no ambos. La ventaja de este sistema consiste en que es necesario sólo un enlace a dos hilos.

La desventaja radica en que el extremo receptor no tiene ninguna forma de avisar al extremo transmisor sobre su estado y sobre la calidad de la información que se recibe. Esta es la razón por la cual, generalmente, no se utiliza.

1.7.1.2. Semi duplex[9]

La comunicación serie se establece a través de una sola línea, pero en ambos sentidos. En un momento el transmisor enviará información y en otro recibirá, por lo que no se puede transferir información en ambos sentidos de forma simultánea.

Este modo permite la transmisión desde el extremo receptor de la información, sobre el estado de dicho receptor y sobre la calidad de la información recibida por lo que permite así la realización de procedimientos de detección y corrección de errores.

1.7.1.3. Full duplex[9]

Se utilizan dos líneas (una transmisora y otra receptora) y se transfiere información en ambos sentidos. La ventaja de este método es que se puede transmitir y recibir información de manera simultánea.

La mayoría de los dispositivos especializados para la comunicación pueden transferir información tanto en full dúplex como en half duplex (el modo simplex es un caso especial dentro de half duplex).

1.7.2. TIPOS DE TRANSMISIÓN[9]

Existen dos formas de realizar la comunicación serial: la sincrónica y la asincrónica, la diferencia entre estas dos formas de comunicación es que la comunicación sincrónica además de la línea para la transmisión de datos, necesita otra línea que contenga los pulsos de reloj, estos a su vez indican cuando un dato es válido.

Por otra parte la comunicación serial asincrónica no necesita de pulsos de reloj, en su lugar utiliza mecanismo como referencia tierra(RS232) o voltajes diferenciales (RS422/485), en donde la duración de cada bit es determinada por la velocidad de transmisión de datos que se debe definir previamente entre ambos equipos.

1.7.2.1. La transmisión asíncrona[9]

Los datos serie se encuentran encapsulados en tramas de la forma:

Figura 1.18: Forma de transmisión asíncrona.

Primero se envía un bit de start, a continuación los bits de datos (primero el bit de mayor peso) y finalmente los bits de STOP.

El número de bits de datos y de bits de Stop es uno de los parámetros configurables, así como el criterio de paridad par o impar para la detección de errores. Normalmente, las comunicaciones serie tienen los siguientes parámetros: 1 bit de Start, 8 bits de Datos, 1 bit de Stop y sin paridad.

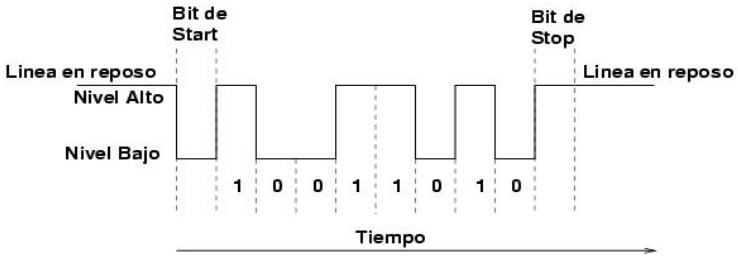


Figura 1.19: Ejemplo de la transmisión del dato binario 10011010. La línea en reposo está a nivel alto.

1.7.2.1.1. Bit de inicio y bit de parada[9]

En la transmisión asíncrona un carácter a transmitir es encuadrado con un indicador de inicio y fin de carácter, de la misma forma que se separa una palabra con una letra mayúscula y un espacio en una oración. La forma estándar de encuadrar un carácter es a través de un bit de inicio y un bit de parada.

Durante el intervalo de tiempo en que no son transferidos caracteres, el canal debe poseer un "1" lógico. Al bit de parada se le asigna también un "1". Al bit de inicio del carácter a transmitir se le asigna un "0". Por todo lo anterior, un cambio de nivel de "1" a "0" lógico le indicará al receptor que un nuevo carácter será transmitido.

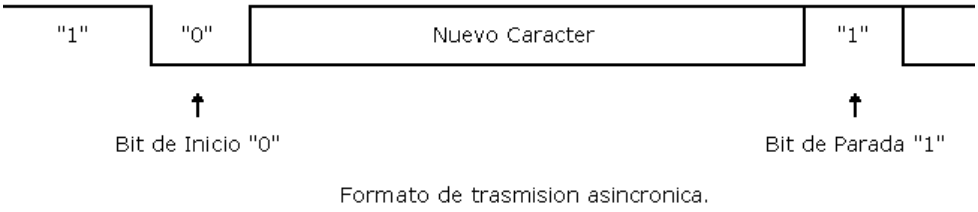


Figura 1.20: Formato de transmision asincrona.

1.7.2.1.2. Reglas de transmisión asíncrona[9]

La transmisión asíncrona que vamos a ver es la definida por la norma RS232, en la que profundizaremos más adelante y que se basa en las siguientes reglas:

a. Cuando no se envían datos por la línea, ésta se mantiene en estado alto (1).
b. Cuando se desea transmitir un carácter, se envía primero un bit de inicio que pone la línea a estado bajo (0) durante el tiempo de un bit.
c. Durante la transmisión, si la línea está a nivel bajo, se envía un 0 y si está a nivel alto se envía un 1.
d. A continuación se envían todos los bits del mensaje a transmitir con los intervalos que marca el reloj de transmisión. Por convenio se transmiten entre 5 y 8 bits.
e. Se envía primero el bit menos significativo, siendo el más significativo el último en enviarse.
f. A continuación del último bit del mensaje se envía el bit (o los bits) del final que hace que la línea se ponga a 1 por lo menos durante el tiempo mínimo de un bit. Estos bits pueden ser un bit de paridad para detectar errores y el bit o bits de stop, que indican el fin de la transmisión de un carácter.

Los datos codificados por esta regla, pueden ser recibidos siguiendo los pasos siguientes:

a. Esperar la transición 1 a 0 en la señal recibida.
b. Activar el reloj con una frecuencia igual a la del transmisor.
c. Muestrear la señal recibida al ritmo de ese reloj para formar el mensaje.

Leer un bit más de la línea y comprobar si es 1 para confirmar que no ha habido error en la sincronización.

1.7.2.1.3. Velocidad de transmisión[9]

En la transmisión asíncrona por cada carácter se envía al menos 1 bit de inicio y 1 bit de parada así como opcionalmente 1 bit de paridad. Esta es la razón de que los baudios no se correspondan con el número de bits de datos que son transmitidos.

.

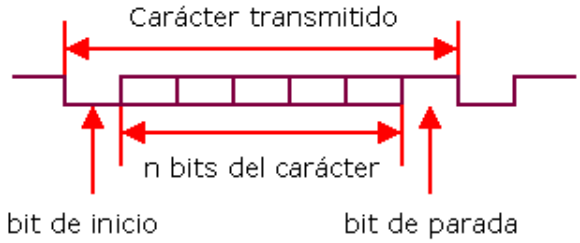


Figura 1.21: Formato básico de transmisión asincrónica.

La característica fundamental del formato de transmisión asíncrono es su capacidad de manejar datos en tiempo real, con un intervalo de longitud arbitraria entre caracteres sucesivos. Al final de cada carácter, la línea va a 1 en el bit de parada y permanece en ese estado durante un número arbitrario de bits ociosos. El inicio del nuevo carácter estará definido por la transición a 0 del bit de inicio.

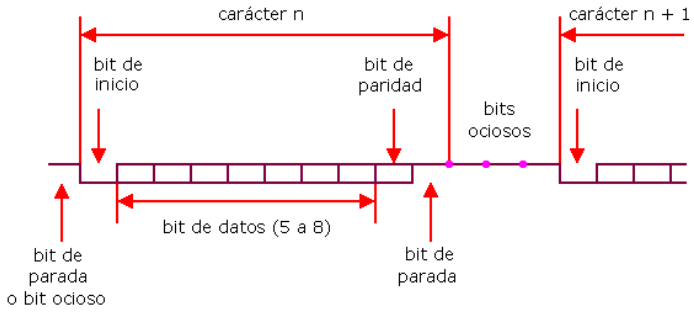


Figura 1.22: Transmisión asincrónica con velocidad menor que la máxima posible.

En la Figura 1.23 se muestra la mayor velocidad asíncrona posible con el bit de paridad.

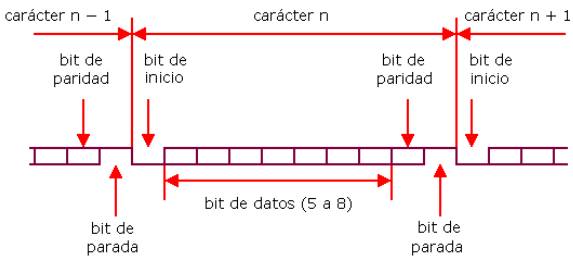


Figura 1.23: Transmisión asincrónica con la velocidad máxima posible.

1.7.2.2. La transmisión síncrona[9]

Es un método más eficiente de comunicación en cuanto a velocidad de transmisión. Ello viene dado porque no existe ningún tipo de información adicional entre los caracteres a ser transmitidos.

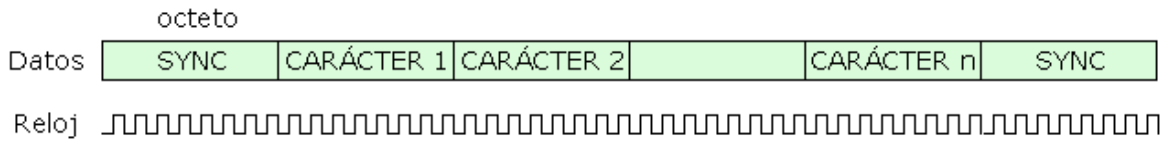


Figura 1.24: Transmisión sincrónica.

Cuando se transmite de manera síncrona lo primero que se envía es un octeto de sincronismo ("sync"). El octeto de sincronismo realiza la misma función que el bit de inicio en la transmisión asíncrona, indicando al receptor que va a ser enviado un mensaje. Este carácter, además, utiliza la señal local de reloj para determinar cuándo y con qué frecuencia será muestreada la señal, es decir, permite sincronizar los relojes de los dispositivos transmisor y receptor. La mayoría de los dispositivos de comunicación llevan a cabo una re sincronización contra posibles desviaciones del reloj, cada uno o dos segundos, insertando para ello caracteres del tipo "sync" periódicamente dentro del mensaje.

Los caracteres de sincronismo deben diferenciarse de los datos del usuario para permitir al receptor detectar los caracteres "sync". Por ejemplo, el código ASCII utiliza el octeto 10010110.

Existen ocasiones en que son definidos dos caracteres de sincronismo, ello puede ser necesario si, por cualquier motivo el carácter "sync" original se desvirtuara, el siguiente permitirá la re inicialización del receptor. En segundo lugar, puede ocurrir que el equipo receptor necesite un tiempo adicional para adaptarse a la señal entrante.

Cuando se transmite de forma síncrona, es necesario mantener el sincronismo entre el transmisor y el receptor cuando no se envían caracteres, para ello son

insertados caracteres de sincronismo de manera automática por el dispositivo que realiza la comunicación.

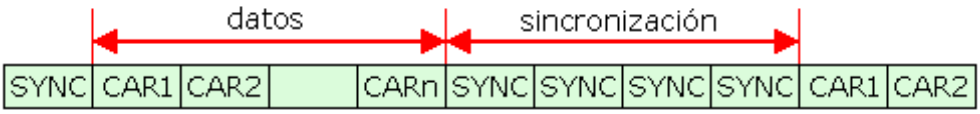


Figura 1.25: Inserción automática de caracteres de sincronismo.

El receptor/transmisor síncrono debe indicar además cuándo el sincronismo ha sido logrado por parte del receptor.

1.7.3. LIMITACIONES DE LA RS-232[10]

La RS-232 tiene una limitación de distancia máxima de 15 metros. Si bien no es una desventaja considerable cuando los equipos a conectar se encuentran cerca, si es un inconveniente cuando la RS-232 se utiliza para conectar directamente terminales que pueden ser lejanas.

Cuando una señal cambia de una condición a otra, la especificación limita el tiempo que puede permanecer en la región de transición. Este requerimiento determina el máximo de capacidad distribuida admisible en el cable, porque la capacidad limita el tiempo de transición de la señal. La norma RS-232 especifica que la capacidad en la línea no debe superar los 2.500 picofaradios.

Los cables se suelen utilizar tienen una capacidad de 120-150 picofaradios por metro de longitud, por lo que la RS-232 tiene como límite de 15m de distancia. Una segunda limitación de la RS-232 es su método de toma de tierra o retorno común. Este método, llamado transmisión no balanceada, funciona bien la mayor parte del tiempo. Sin embargo, si hay diferencia de potencial entre los dos extremos del cable (lo cual es bastante probable en recorridos largos), se reduce

[10] http://www.museo8bits.com/wiki/index.php?title=RS-232

la región de transición entre marca y espacio. Cuando ocurre esto existe la posibilidad que no se interpreten bien los distintos estados de la señal.

1.7.4. DESCRIPCIÓN DE TERMINALES EN RS-232[11]

El Rs-232 consiste en un conector tipo DB-9 de 9 pines o dB-25 de 25 pines. Las señales con las que trabaja este pórtico serie son digitales, así los niveles de salida son de -5 a -15 voltios para el 1 lógico y +5 a +15voltios para el 0 lógico, mientras que los niveles de entrada son -3 a -15 voltios para un 1 lógico y +3 a +15 voltios para un 0 lógico.

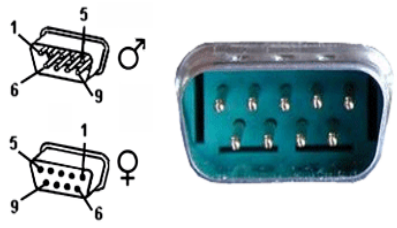


Figura 1.26: Conector db9.

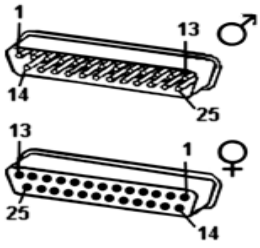


Figura 1.27: Conector db-25.

Cada pin puede ser de entrada o de salida, teniendo una función específica cada uno de ellos. Las señales TXD, DTR y RST son de salida, mientras que RXD, DSR, CTS y DCD son de entrada como se muestra en la siguiente tabla. La masa de referencia para todas las señales SG (Tierra de Señal).

1.7.4.1. Pines del puerto serie (Conexiones D25 y D9)

Nº de PIN DB-25	Nº de PIN DB-9	Abreviación	Nombre completo
2	3	TD	Tansmit Data
3	2	RD	Receive Data
4	7	RTS	Request to send

[11] http://perso.wanadoo.es/pictob/comserie.htm#rs232

5	8	CTS	Clear to send
6	6	DRS	Data set ready
7	5	SG	Signal ground
8	1	CD	Carrier detect
20	4	DTR	Data terminal ready
22	9	RI	Ring indicator

Tabla 1.1: Pines del puerto serie (conexiones d25 y d9).

1.7.5. FUNCIONES DE LOS PINES

Abreviación	**Nombre completo**	**Función**
TD	Transmita Data	Salida de datos Serie (TXD)
RD	Receive Data	Entrada de datos Serie (RXD)
CTS	Clear to Send	Indica si el modem está preparado para recibir y mandar datos
DCD	Data Carrier Detect	Detecta si hay conexión con el otro extremo
DSR	Data Set Ready	Dice si la UART está preparada para la conexión
DTR	Data Terminal Ready	Dice al modem si la UART está preparada para la conexión
RTS	Request To Send	Informa que la UART está preparada para intercambiar datos
RI	Ring Indicator	Se activa cuando el modem detecta una llamada del PSTN

Tabla 1.2: Funciones de los pines.

1.7.6. CONVERSOR TTL-RS232[11]

Los niveles de +/- 12V de la interface RS-232 de una PC no son compatibles con los niveles TTl de la mayoría de los microcontroladores o de otras aplicaciones como agendas electrónicas o celulares. Una solución es el integrado MAX232.

Los puertos RS232 usan voltajes bipolares de +/- 12V para representar los unos y los ceros, mientras que en TTl se usa 0 y 5V. Un popular integrado para hacer esta conversión es el max232 que usa algunos condensadores externos para generar estos voltajes a partir de una fuente de 5V.

1.7.7. MAX 232[12]

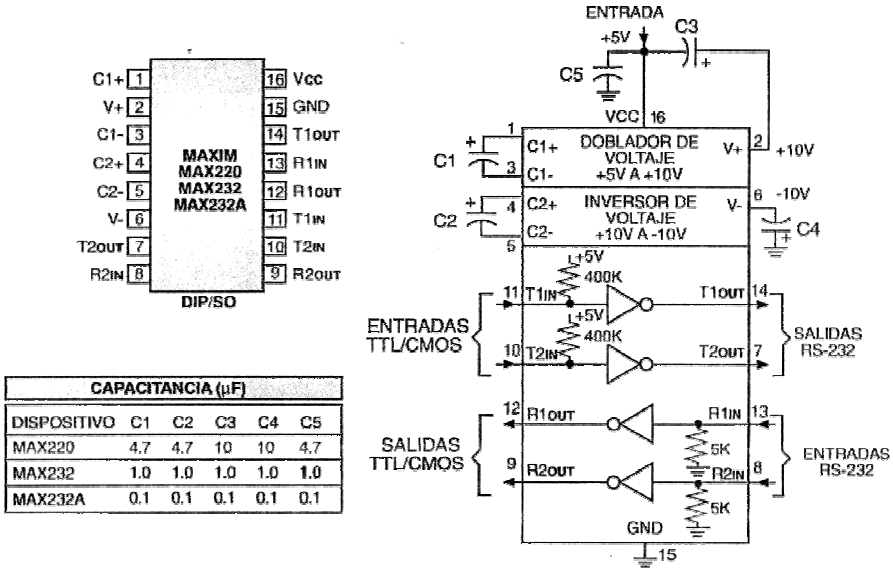
Este circuito integrado soluciona los problemas de niveles de voltaje cuando se requiere enviar señales digitales sobre una línea RS-232. El MAX232 se usa en aquellas aplicaciones donde no se dispone de fuentes dobles de +/-12 voltios; por ejemplo, en aplicaciones alimentadas con baterías de una sola polaridad. El MAX232 necesita solamente una fuente de +5 voltios para su operación; un elevador de voltaje interno convierte el voltaje de +5 V al de doble polaridad de +/- 12V.

Como la mayoría de las aplicaciones de RS-232 necesitan de un receptor y un emisor, el MAX232 incluye en un solo paquete 2 parejas completas de driver y receiver, como lo ilustra la estructura interna del integrado que se muestra en la Figura 1.28.

El MAX232 tiene un doblador de voltaje de +5V a +10voltios y un inversor de voltaje para obtener la polaridad de -10V. El primer convertidor utiliza el condensador C1 para doblar los +5V de entrada a +10V sobre el condensador C3 en la salida positiva V+. El segundo convertidor usa el condensador C2 para invertir de +10V a -10V en el condensador C4 de la salida V-. El valor mínimo de

[12] http://robots-argentina.com.ar/Comunicacion_max232.htm

estos condensadores los sugiere el fabricante, aunque en la práctica casi siempre se utilizan condensadores de Tantalio de 10µF.



CAPACITANCIA (µF)					
DISPOSITIVO	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figura 1.28: Diagrama de pines y estructura interna MAX232.

Una aplicación clásica consiste en conectar las salidas para transmisión serial TX y Rx de un microcontrolador a una interface RS-232 con el fin de intercambiar información con una computadora. La mayoría de los sistemas concentrados de datos están compuestos por sensores conectados a microcontroladores que, a su vez, vía RS-232 le comunican los datos recolectados a un computador central. El MAX232 implementa la interface con la misma fuente de alimentación de +5 voltios. En la siguiente Figura 1.29 se ilustra la conexión serial de un microcontrolador a través del MAX232.

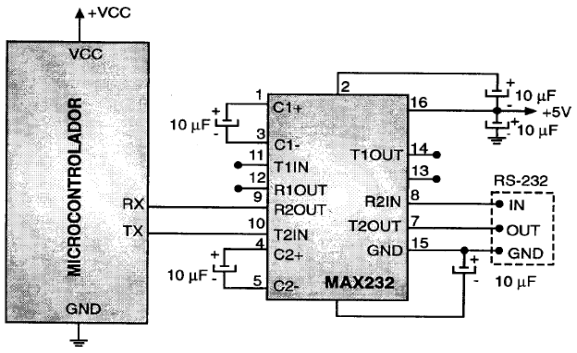


Figura 1.29: Aplicación típica del max232.

1.7.7.1 Características del max232

Tabla: 1.3: Características del Max 232.

LIMITES			
Fuente de alimentación	-0.3 a+6V.		
Voltajes de entrada:			
Tin	-0.3v A (Vcc-0.3V)		
Rin	+/-30V		
Voltajes de salida:			
Tout	+/-15V		
Rout	-0.3V a (Vcc +0.3V)		
Protección Corto	Continua		
Disipación de potencia	842mW		
CARACTERISTICAS A Vcc=+5V, C1-C4=0.1µF			
	Min	**Tip.**	**Máx.**
TRANSMISOR			
Voltaje de salida(carga 3KΩ)	+/-5V	+/-8V	
Entrada BAJA		1.4V	0.8V
Entrada BAJA	2	1.4 V	
Velocidad		200Kb/seg.	
RECEPTOR			
Rango de entrada			+/-30V
Entrada BAJA	0.8V	1.3V	
Entrada ALTA		1.8V	2.4V
Resistencia de entrada	3KΩ	5KΩ	7KΩ

CAPÍTULO II:
FUNDAMENTOS TEÓRICOS RESPECTO AL SOFTWARE

2.1. VISUAL BASIC[13]

2.1.1. INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS[13]

La programación orientada a objetos es aquella en la que trabajamos con objetos visibles, cada uno de los cuales posee sus propias características, métodos y eventos. Los objetos son, por ejemplo, un botón de comando, una caja de texto, una imagen, un botón de opción, una lista desplegable, una barra de desplazamiento, en general todo objeto visible que usted puede observar en la pantalla.

2.1.2. PROGRAMAS ORIENTADOS A EVENTOS[13]

Todas las aplicaciones creadas en un lenguaje de programación orientado a objetos serán por defecto un programa orientado a evento. Es un programa orientado a eventos, porque cuando este es cargado o ejecutado solo espera a que el usuario realice alguna acción sobre uno de los objetos que posee, por ejemplo, la calculadora de Windows espera a que el usuario haga clic (Evento Click) con el Mouse sobre uno de los botones de comando que contienen los números para luego ponerlo en la caja de texto.

2.1.3. CREACIÓN DE PROGRAMAS PARA EL ENTORNO DE WINDOWS [13]

En Visual Basic 6.0 es posible crear todo tipo de aplicación para Windows, pudiendo incorporar todas las características y elementos de un programa típico de Windows.

---

[13] http://www.scribd.com/doc/11493749/Libro-de-ORO-de-Visual-Basic-60-Orientado-a-Bases-de-Datos-2da-Ed

Visual Basic 6.0 posee una barra de herramientas donde es posible encontrar todos los elementos que apreciamos en una aplicación de Windows: ventanas, botones, cajas de texto, cajas de diálogo, botones de opción, botones de selección, barras de desplazamiento, gráficos, menús, en general, todo objeto visible en una aplicación de Windows.

2.1.4. ELEMENTOS DE UNA APLICACIÓN EN WINDOWS[13]

Con Visual Basic es mucho más fácil crear aplicaciones para Windows sin la preocupación de tantas definiciones de variables, constantes y punteros. Los elementos principales de una aplicación de Windows son: formularios, controles, procedimientos, métodos, objetos y eventos. Cada uno de estos elementos juega un papel muy importante en una aplicación de Windows.

2.1.5. FORMULARIOS O VENTANAS[13]

En Visual Basic, un formulario es considerado como una ventana típica de Windows. En este se colocan todos los controles de los que dispone Visual Basic para crear una aplicación. Una aplicación puede tener uno o varios formularios, pero un único formulario puede ser suficiente para la creación de una potente aplicación. El número de formularios va a depender precisamente del tipo de aplicación y de la cantidad de módulos de trabajo que incluirá la aplicación.

2.1.5.1. Nombre de controles y objetos

Cada control u objeto en Visual Basic debe tener un nombre, por medio del cual se puede hacer referencia a dicho objeto en la aplicación. El nombre (name), puede ser el que el programador desee, e incluso Visual Basic proporciona nombres por defecto para los diversos controles. Estos nombres por defecto, hacen referencia al tipo de control y van seguidos de un número, que se incrementa a medida que se van introduciendo más controles de ese mismo tipo en el formulario; por ejemplo, Text1 para una caja de texto, Text2 para otra caja de texto, Command1 para un botón de comando, Command2 para otro botón de comando, así sucesivamente.

2.1.5.2. Propiedades, clase y estado de objetos

Se ha dicho que los objetos que colocamos en un formulario de Visual Basic, también son llamados controles, cada uno de los cuales posee propiedades, métodos y eventos. Las propiedades son las características propias de un objeto
El estado de un objeto puede cambiar en Visual Basic, por ejemplo si un objeto tiene un color puede cambiar a otro color.

Casi todas las propiedades de los controles de Visual Basic pueden cambiarse en momento que la aplicación se está diseñando (modo de diseño), y también casi siempre cuando la aplicación esta en ejecución (modo de ejecución). Para modificar u obtener el estado de un objeto se hace por medio del nombre del objeto (Name), seguido de un punto (.) y el nombre de la propiedad. Por ejemplo, para cambiar el color de una caja de texto llamada Text1, se haría de la siguiente manera:

Text1.BackColor = vbRed: Donde Text1 es el nombre del objeto, BackColor el nombre de laPropiedad y vbRed es el color rojo por defecto de Visual Basic. En Visual Basic los colores se representan por constantes y valores hexadecimales

Por otro lado, la clase representa la entidad genérica a la que pertenece un objeto, por ejemplo, en una aplicación, puede haber varios botones de comando, cada uno de los cuales es un control que pertenece a una clase de objetos, llamada CommandButton (botones de comando). La clase del objeto también determina las propiedades de los objetos de esa clase, es decir, cada clase, tipo de objeto o control tienen su conjunto de propiedades, y cada objeto o control tienen valores determinados para las propiedades de su clase.

2.1.6. MODO DE DISEÑO Y MODO DE EJECUCIÓN[13]

Las aplicaciones en Visual Basic pueden trabajar en dos modos distintos, que son: modo de diseño y modo de ejecución. En modo de diseño, el programador construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades y codificando los procedimientos para gestionar los eventos de cada control. La aplicación se prueba en modo de ejecución, en este caso, el programador actúa sobre la aplicación produciendo los eventos codificados y obteniendo los valores proporcionados por dichos eventos. Para que una aplicación pase del modo de diseño al modo de ejecución simplemente hay que pulsar la tecla [F5], o bien, hacer clic en el botón Iniciar (Start) , de la barra de herramientas estándar.

2.1.7. PROCEDIMIENTOS[13]

Un procedimiento representa la definición o declaración de un objeto. Los procedimientos se encuentran dentro de los módulos, y contienen el conjunto de instrucciones que se ejecutan cuando el usuario realiza algún evento sobre el objeto, o cuando estos son llamados por su nombre. Para definir un procedimiento, se utiliza la palabra clave Sub, seguida del nombre del procedimiento y los argumentos (si el procedimiento lo requiere). Para declarar un procedimiento como privado se utiliza la palabra clave Private y para declarar un procedimiento como público se utiliza la palabra clave Public.

2.1.8. EVENTOS[13]

Las acciones que realiza el usuario sobre un objeto se llaman eventos. Son eventos típicos: hacer clic sobre un botón (evento Click), hacer doble clic sobre un fichero (evento DblClick), arrastrar un icono (evento DragOver), pulsar una tecla o una combinación de teclas (evento KeyPress), escribir en una caja de texto (evento Change), o simplemente desplazar el puntero del Mouse sobre un objeto

(evento MouseMove). Los eventos vienen especificados junto con el procedimiento del objeto, separado por el carácter underscore o subrayado (_).
Por ejemplo, el evento Click de un botón de comando vendría representado de la siguiente manera:

```
Private Sub Command1_Click ( )
End ‘La sentencia End se utiliza para salir de la aplicación.
End Sub
```

La Tabla 2.1 muestra los eventos más comunes de los controles de Visual Basic 6.0.

Evento	Descripción
Click	Ocurre cuando el usuario presiona y suelta un botón del mouse sobre un *objeto*.
DblClick	Ocurre cuando el usuario presiona y suelta dos veces un botón del mouse sobre un *objeto.*
DragDrop	Ocurre como resultado de arrastrar y soltar con el mouse un *control* sobre un determinado *tipo de objeto.*
DragOver	Ocurre cuando una operación de arrastrar y colocar está en curso. Puede usar este *evento* para controlar el puntero del mouse a medida que entra, sale o descansa directamente sobre un destino válido.
GotFocus	Ocurre cuando un *objeto* recibe el { CONTROL Internet.HHCtrl.1 }{ HYPERLINK "JavaScript:alink_4.Click()" }, ya sea mediante una acción del usuario, como tabular o hacer clic en el objeto, o cambiando el enfoque en el código mediante el método **SetFocus**.
LostFocus	A diferencia del evento anterior, este *evento* ocurre cuando el *objeto* pierde el enfoque, ya sea mediante tabulaciones o hacer clic sobre otro *objeto*.
KeyDown	Ocurre cuando el usuario mantiene presionada una tecla.
KeyUp	Ocurre cuando el usuario termina la operación de pulsar una tecla. Se podría decir, que este evento ocurre precisamente al terminar el evento **KeyDown**.
KeyPress	Ocurre como resultado de presionar y soltar una tecla.
MouseDown	Ocurre cuando el usuario presiona un botón del mouse, pero a diferencia del *evento* **MouseDown**, permite identificar cuales de los tres botones del mouse fue presionado y las combinaciones de tecla **ALT**, **MAYÚS** y **CTRL**.
MouseUp	El evento **MouseUp** se produce cuando el usuario suelta el botón del mouse. **MouseUp** es un compañero útil a los eventos **MouseDown** y **MouseMove**.
MouseMove	Este evento ocurre mientras el usuario mueve o desplaza el puntero del mouse sobre un *objeto.*

Tabla 2.1: Eventos más comunes de los controles de Visual Basic 6.0.

2.1.8. MÉTODOS[13]

Los métodos son funciones que también son llamadas desde el programa, pero a diferencia de los procedimientos no son codificados por el programador. Los métodos, son llamados desde código en Visual Basic de la misma manera como se hace referencia a una propiedad del control. La Tabla 2.2 muestra los métodos más comunes de los controles de Visual Basic 6.0:

Método	Descripción
Drag	Inicia, termina o cancela una operación de arrastre de cualquier *control*, excepto los *controles* ***Line, Menu, Shape, Timer o CommonDialog.***
Move	Se utiliza para mover un *control o formulario*, especificando sus coordenadas ***(Top, Left)*** y su tamaño ***(Width, Height)***.
Refresh	Se utiliza para dibujar o actualizar gráficamente un *control* o un *formulario*. Se utiliza principalmente con los *controles* **FileListBox** y **Data**.
SetFocus	Este *método* se utiliza para hacer que un *objeto* reciba el enfoque. Este *método* es uno de los más usados para los *controles* de *Visual Basic 6.0*.
ShowWhatsThis	Permite mostrar un tema seleccionado de un archivo de Ayuda utilizando el menú emergente **¿Qué es esto?** que ofrece la ayuda de Windows. Este *método* es muy útil para proporcionar ayuda interactiva en un menú contextual acerca de un *objeto* en una aplicación. Este *método* muestra el tema indicado por la propiedad **WhatsThisHelpID** del *objeto* especificado en la sintaxis.
Zorder	Se utiliza para que un *control* o *un objeto formulario* se coloque por encima o por debajo de otros *objetos*.

Tabla 2.2: Métodos más comunes de los controles de Visual Basic 6.0.

2.1.9. MÓDULOS[13]

Un módulo se puede definir, como el lugar donde se almacena el código fuente de una aplicación en Visual Basic. Los módulos se clasifican en tres tipos que son: formulario, estándar y de clase.

MÓDULOS DE FORMULARIO: Los módulos de formulario se almacenan en un archivo con extensión .frm, y son la base de la mayoría de las aplicaciones de Visual Basic. Pueden contener procedimientos que controlen eventos, procedimientos generales y declaraciones a nivel de formulario de variables, constantes, tipos y procedimientos externos.

MÓDULOS ESTÁNDAR: Los módulos estándar trabajan independientemente de la aplicación y se almacenan en un archivo con extensión .bas. En ocasiones, el programador necesitará declarar módulos generales, que contengan códigos que puedan ser utilizados desde varios formularios, para así evitar tener que volver a repetir el código. Pueden contener declaraciones disponibles para toda la aplicación o a nivel de módulo de variables, constantes, tipos, procedimientos externos y procedimientos globales.

MÓDULOS DE CLASE: Los módulos de clase, son la base de la programación orientada a objetos en Visual Basic y se almacenan en archivos con extensión .cls. Puede escribir código en módulos de clase para crear nuevos objetos. Estos objetos nuevos pueden incluir propiedades y métodos personalizados. En realidad, los formularios sólo son módulos de clase que pueden tener controles y que pueden mostrar ventanas de formulario.

2.1.10. ENCAPSULACIÓN, HERENCIA, POLIMORFISMO Y MENSAJES EN OBJETOS[13]

Encapsulación: Los objetos o controles poseen códigos y estructuras internas ocultas para otras entidades. La capacidad de ocultar y aislar el código de un objeto o control en una aplicación se denomina Encapsulación.

Herencia: Existen objetos que adquieren o heredan propiedades y métodos de otros objetos de mayor jerarquía, ya sea de clases superiores o una superclase. Esta capacidad de compartir la estructura de otros objetos de clases superiores se denomina Herencia.

Polimorfismo: Significa que muchas clases pueden proporcionar la misma propiedad o el mismo método, y que el objeto que llama no tiene por qué saber la clase a la que pertenece el objeto que recibe la solicitud antes de ser invocada una propiedad o un método de dicho objeto.

Mensajes en objetos: Los mensajes en objetos no son más que las llamadas a los métodos de dicho objeto.

2.1.11. EL EID DE VISUAL BASIC 6.0[13]

El EID de Visual Basic 6.0 se muestra en la Figura 2.1, posee un sin número de herramientas que hacen de la programación de aplicaciones en Visual Basic mucho más fácil e interactiva. Este entorno incluye elementos tales como: barra de menús, barra de controles, barra de herramientas, ventana de propiedades, ventana de proyectos, depurador, formularios, etc.

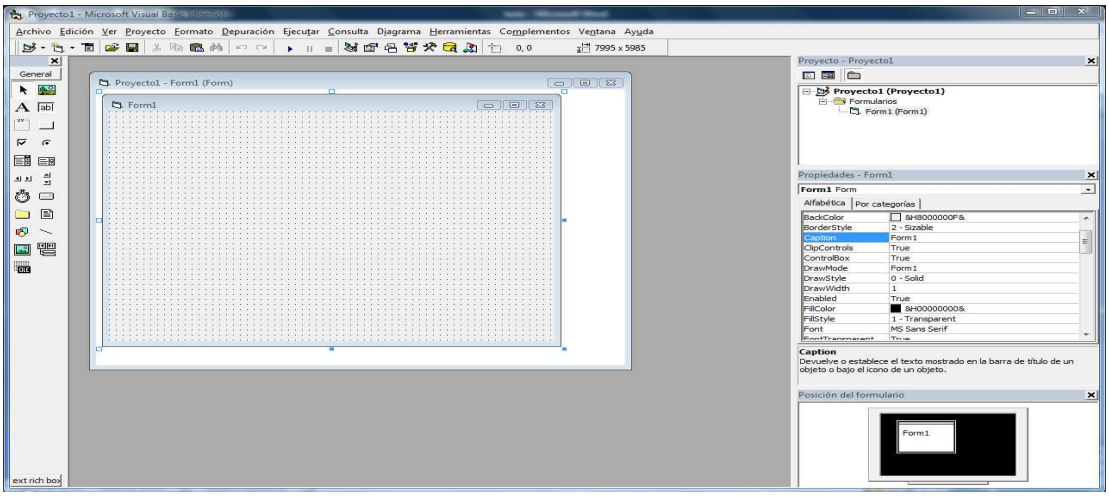


Figura 2.1: El EID de Visual Basic 6.0.

A continuación los siguientes elementos:

2.1.11.1. La barra de menús

La barra de menús de Visual Basic 6.0 resulta muy similar a la de cualquier otra aplicación de Windows, tal y como se puede apreciar en la Figura 2.2. Algunos de los menús de esta barra tienen muy poca novedades, es decir, algunos incluyen las opciones típicas de los menús de cualquier aplicación de Windows.

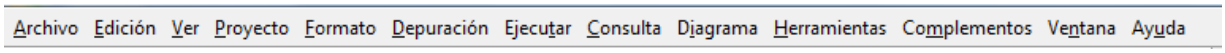


Figura 2.2: La barra de menús.

El menú **Archivo** tiene pocas novedades. Lo más importante es la distinción entre proyectos. Un proyecto reúne y organiza todos los ficheros que componen el programa o aplicación.

El menú **Edición** aporta cambios importantes sobre lo que es lo habitual.

El menú **Ver**, generalmente de poca utilización, es bastante propio de Visual Basic 6.0. Este permite hacer aparecer en pantalla las distintas ventanas del entorno de desarrollo, así como acceder a un formulario o al código relacionado con un control, y manejar funciones y procedimientos.

El menú **Proyecto** permite añadir distintos tipos de elementos a un proyecto.
Con ProjectProperties se puede elegir el tipo de proyecto y determinar el formulario con el que se arrancará la aplicación (Startup Object).

En menú **Herramientas** se encuentran los comandos para arrancar el Menu Editor y para establecer opciones del programa.

En Complementos se encuentran una serie de opciones que permiten configurar el EID de Visual Basic 6.0. En páginas siguientes se verá con más detalles los elementos que componen la ventana Options del EID de Visual Basic 6.0.

Por último, la Ayuda (Help) (siempre imprescindible y en el cado de Visual Basic 6.0 particularmente muy bien hecha) que se encuentra en el menú Help, se basa fundamentalmente en una clasificación temática ordenada de la información disponible (Contents), en una clasificación alfabética de la información (Index) y en la búsqueda de información sobre algún tema por el nombre (Search).

2.1.11.2. La barra de herramientas estándar

La Barra de Herramientas Estándar aparece en la Figura 2.2 permite acceder a las opciones más importantes de los menús de Visual Basic.

Figura 2.2: La Barra de Herramientas Estándar.

Esta barra posee algunos elementos típicos de Windows como: nuevo, abrir, guardar, copiar, cortar, pegar, buscar, deshacer y rehacer, aunque también posee elementos que son exclusivos del EID de Visual Basic.

2.1.11.3. La caja de herramientas (Toolbox)

En la Figura 2.3 muestra la caja de herramientas, que incluye los controles con los que se puede diseñar la pantalla de la aplicación. Estos controles son por ejemplo, botones de comando, etiquetas, cajas de texto, imágenes, etc.

Figura 2.3: Caja de herramientas (Toolbox).

El número de controles que pueden aparecer en esta ventana varían con la configuración del sistema, pero usted puede agregar nuevos componente a la caja de controles.

2.1.11.4. Los formularios (Forms)

Los formularios son las zonas de la pantalla sobre las que se diseña el programa y sobre los que se sitúan los controles o herramientas de la Caja de Herramientas, como podemos observar en la Figura 2.4. Al ejecutar el programa, el Formulario se convertirá en la ventana principal de la aplicación, donde aparecerán los botones, las cajas de texto, los gráficos, etc.

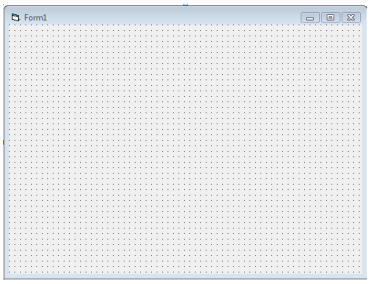


Figura 2.4: Caja de herramientas (Toolbox)

2.1.11.5. El explorador de proyectos (Project Explorer)

El explorador de proyectos visualiza lo distintos formularios y módulos que componen un proyecto, Figura 2.5. Estos módulos están representados por un icono que lo diferencian de los demás módulos. Para visualizar uno de estos módulos o formularios, solo debe hacer doble click en el icono que lo representa. Si usted no ve el explorador de proyectos, entonces, pulse la combinación de teclas Ctrl+R.

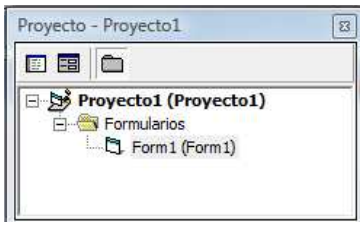


Figura 2.5: Explorador de proyectos.

2.1.11.6. La ventana de propiedades (Properties Windows)

En la ventana de propiedades Figura 2.6, se muestran todas las propiedades de un control o un módulo seleccionado en la aplicación. Mediante esta ventana es posible cambiar los valores de las propiedades de cada uno de los controles, indicando así, sus características y estados antes de ejecutar la aplicación.

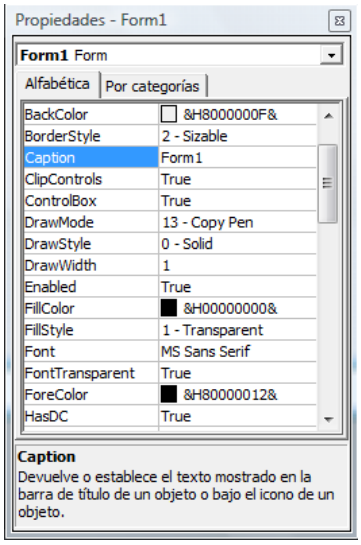


Figura 2.6: Explorador de proyectos.

2.1.11.7. La ventana de esquema de formularios (Form Layout Window)

La ventana de la Figura 2.6 permite observar y alinear en la pantalla cada uno de los formularios de una aplicación, es decir, permite indicar la posición donde debe de aparecer cada uno de los formularios cuando la aplicación este corriendo o se esté ejecutando.

Figura 2.6: Explorador de proyectos.

2.1.11.8. La ventana explorador de formulario (Form Explorer Window)

La ventana explorador de formulario (Form Explorer Windows) muestra el formulario con el cual se esta trabajando en la aplicación. Esta ventana aparece por defecto cada vez que se carga una aplicación y es la única ventana que permite al usuario interactuar con los formularios y controles que se encuentran en la aplicación. La Figura 2.7 muestra la ventana explorador de formularios:

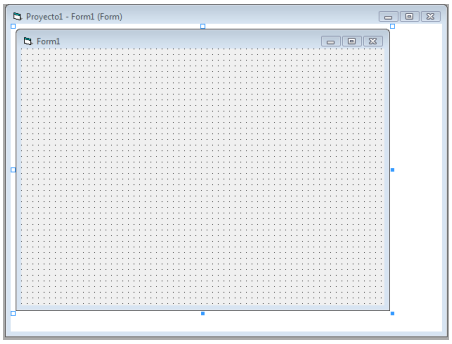


Figura 2.7: Explorador de formularios.

2.1.11.9. El Editor de Código (Code Editor)

El Editor de Código de Visual Basic 6.0 es la ventana en la cual se escriben las sentencias de los procedimientos y módulos de la aplicación. Esta ventana se activa de formas diferentes, una de las principales es haciendo doble clic sobre un formulario o sobre cualquiera de sus controles. La siguiente Figura 2.8 muestra el aspecto físico del Editor de Código:

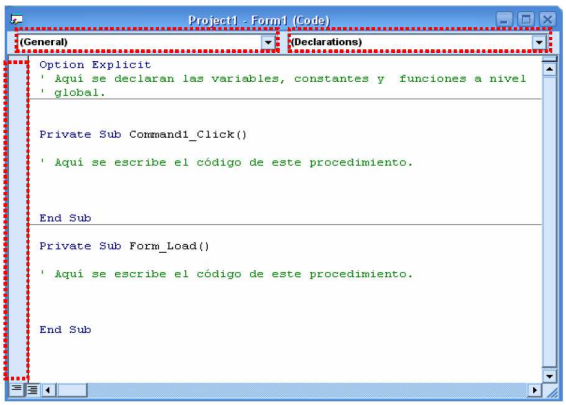


Figura 2.8: Editor de código.

El Depurador (Debugger)

El Depurador es una herramienta utiliza para la corrección y detección de errores en la aplicación. Hoy en día la mayoría de los entornos de programación poseen potentes herramientas que facilitan la depuración de los programas realizados. La característica principal del Depurador es que permite ejecutar parcialmente el programa, deteniendo la ejecución en una línea de código determinada. Esta barra se muestra en la siguiente Figura 2.9:

Figura 2.9: Depurador.

2.1.12. LOS CONTROLES MÁS USUALES EN VISUAL BASIC 6.0[13]

Los controles más usuales son aquellos que usamos con mayor frecuencia al momento de crear una aplicación. Estos controles son por ejemplo: botones de comando, botones de opción, cajas de texto, etiquetas, barras de desplazamiento, listas, cajas combinadas, botones de verificación, etc.

2.1.12.1. Los botones de comando (CommandButton)

Los botones de comando son aquellos botones típicos que vemos frecuentemente en las aplicaciones de Windows, ver Figura 2.10, que realizan una operación en específico, por ejemplo, salir, imprimir, cancelar, etc.

Figura 2.10: Botones de comando.

Los botones de comando se utilizan principalmente para iniciar, interrumpir o terminar un proceso.

2.1.12.1.1. Propiedades de los botones de comando

Propiedad	Descripción
Name	Se utiliza para asignarle el nombre al control.
BackColor	Cambia el color del botón de comando. Para que el botón tome el color seleccionado en la propiedad BackColor, usted debe establecer el valor "1- Graphical" en la propiedad Style del botón de comando.
Caption	Establece el texto que aparece escrito sobre el objeto, en este caso sobre el botón de comando.
Enabled	Habilita o deshabilita el objeto, es decir, indica si el objeto responderá a los eventos del usuario.
Font	Permite cambiar el tipo de fuente del texto.
Height y Width	Permite cambiar la altura y anchura del objeto expresada en Twips (unidad de medida de la pantalla)
Left y Top	Permite cambiar la posición a la izquierda y superior del control sobre su contenedor.
Picture	Asigna una imagen (Bitmap) en el objeto. En un botón de comando esta propiedad tendrá efecto siempre y cuando el valor de la propiedad Style este establecido a "1 – Graphical".
Visible	Establece si el control estará Visible cuando se ejecute la aplicación si el valor esta True, entonces el botón de comando estará visible, de lo contrario, estará Invisible.
ToolTipText	Se utiliza para mostrar el texto contextual que aparece cuando se coloca el puntero del mouse sobre el objeto o control.

Tabla 2.3: Propiedades de los botones de comando

2.1.12.2. Las etiquetas (Labels)

Una Etiqueta o Label es un objeto gráfico que se coloca en una parte determinada de un contenedor para mostrar un texto que no puede ser modificado directamente por el usuario.

2.1.12.2.1. Propiedades de las etiquetas

A continuación, algunas de propiedades más utilizadas de las Etiquetas:

Propiedad	**Descripción**
Caption	Se utiliza para mostrar un texto sobre un objeto o control colocado sobre un contenedor (Formulario, Frame, etc).
AutoSize	Con esta propiedad una etiqueta se ajusta al tamaño del texto escrito en la propiedad Caption.
BackColor	Establece el color de fondo de la Etiqueta.
BackStyle	Esta propiedad especifica si la Etiqueta tendrá color de fondo o si totalmente transparente.
Font	Es utilizada esta propiedad para cambiar el tipo de fuente del texto.
ForeColor	La propiedad ForeColor devuelve o establece el color del texto contenido en un control.

Tabla 2.3: Propiedades de las etiquetas

2.1.12.3. Las cajas de texto (TextBox)

Un control TextBox, llamado también caja de edición, es aquella que permite al usuario introducir datos en tiempo de ejecución.

2.1.12.3.1. Propiedades de las cajas de texto

A continuación las propiedades más relevantes:

Propiedad	Descripción
Alignment	Esta propiedad permite justificar el texto que el usuario ingresara por el teclado o por medio de otra fuente de información.
Appearance	Establece si la caja de texto tendrá o no apariencia 3D. Esta propiedad puede tomar los valores 0 – Flat (sin bordes 3D) y 1 – 3D (con bordes 3D), este último es el valor por defecto.
BorderStyle	Esta propiedad devuelve o establece el estilo de borde de la caja de texto, es decir, indica si la caja de texto tendrá o no bordes en sus extremos.
Locked	Devuelve o establece un valor que indica si la caja de texto se puede modificar.
Text	Todo lo que sea escrito en la caja de texto se almacena en la propiedad Text en tiempo de ejecución.
MaxLength	Devuelve o establece un valor numérico que indica la cantidad máxima de caracteres que puede aceptar una caja de texto en el área de edición. El valor máximo para una caja de texto es de aproximadamente 65,535 caracteres o 32 KB.
MultiLine	Devuelve o establece un valor que indica si la caja de texto admitirá más de una línea de texto en el área de edición.
ScrollBars	Devuelve o establece un valor que indica si la caja de texto tendrá barra de desplazamiento horizontal o vertical.
PasswordChar	Devuelve o establece un valor que se muestra cada vez que se escribe en la caja de texto. Esta propiedad solo admite un carácter. Es utilizada para ocultar los datos que se escriban en la caja de texto. Los caracteres escritos se sustituyen por el carácter especificado en la propiedad PasswordChar.

Tabla 2.3: Propiedades de la caja de texto

2.1.12.4. El control tiempo (Timer)

Un control Timer puede ejecutar código a intervalos periódicos produciendo un evento Timer, que ocurre cuando ha transcurrido un Intervalo preestablecido para un control Timer. La frecuencia del intervalo se almacena en la propiedad Interval del control que especifica el tiempo en milisegundos, ver la Figura 2.11:

Figura 2.11: Explorador de formularios.

2.1.12.4.1. Propiedades del control tiempo

De todas las propiedades de un control tiempo las más importantes son Enabled para habilitar o deshabilitar el control y la propiedad Interval para especificar el intervalo de tiempo en que el control realizará el evento.
Para representar los segundos en milisegundos solo debe multiplicar la cantidad de segundos por mil. Por ejemplo, 2 segundos sería 2 x 1000 = 2000 milisegundos, un minuto sería 60 * 1000 = 60000 milisegundos, así sucesivamente.

2.1.13. INTRODUCCIÓN AL LENGUAJE BASIC[13]

Todos los lenguajes de programación están compuestos por una serie de objetos que hacen posible su funcionamiento entre los cuales tenemos: variables, constantes, tipos de datos, sentencias, expresiones, funciones y estructuras.

2.1.13.1. Identificadores

Los identificadores representan una expresión que hace referencia a una variable o una constante. Un identificador es una secuencia de caracteres que puede tener una longitud máxima de 255 caracteres.

2.1.13.2. Palabras reservadas

Las palabras reservadas del lenguaje Basic no se pueden utilizar como identificadores, ya que tienen significado especial en Visual Basic y no se utilizan para otros propósitos. A continuación, se muestra la Figura 2.12 donde hay algunas palabras reservadas del lenguaje Basic:

Left	Let	Like	Loc
Lock	Lof	Log	Loop
Lset	Ltrim	Me	Mid
Minute	Mirr	Mkdir	Mod
Month	Name	New	Next
Not	Nper	Npv	Oct
On	Onerror	Open	Or
Option	Print	Ppmt	Print#
Private	Property	Public	Put
Pv	Qbcolor	Raise	Randomize
Rate	Redim	Remove	Reset
Resume	Return	Rgb	Right
Emdir	Rnd	Rset	Rtrim
Savesettings	Second	Selectcase	Seek
Shell	SendKeys	Setattr	Sgn
Sin	Single	Sin	Space
Spc	Sqr	Static	Stop
Str	Strcomp	Strconv	String
Sub	Syd	Swich	Tab
Tan	Timer	TimeSerial	TimeValue
Trim	Typename	Ubound	Ucase
Unlock	Val	Vartype	Weekday
Wend	While	Width	Write#
Xor	Yeqr	#if	#else

Figura 2.12: Palabras reservadas.

2.1.13.3. Tipos de datos

Los tipos de datos son los distintos objetos de información con los que trabaja una aplicación en Visual Basic. Todos los datos tienen un tipo asociado con ellos. Un dato puede ser un simple carácter como un “B”, una cadena de caracteres como “La casa de pedro”, un valor entero como 242, un número real como 3.1415 o un valor lógico como True o False.

2.1.13.3.1. Clasificación de los tipos de datos

Cada tipo de información tiene un tipo de datos asociados, cualquier tipo de datos estará definido dentro de la siguiente clasificación: tipos enteros (Byte, Integer, Long), tipos reales (Single, Double, Currency), tipos cadena (String), tipos lógicos (Boolean), tipos fecha (Date) y tipos variados (Variant). Al seleccionar un tipo de datos adecuado para las variables de su programa ahorraría mucho espacio en la memoria del computador donde se ejecute la aplicación y como resultado esta trabajaría mucho más rápido. El tipo datos a seleccionar va a depender del tipo de información que usted valla a almacenar en una variable. A continuación, se muestran todos los tipos de datos disponibles en Visual Basic y el tipo de información que cada uno de ellos pueden almacenar, así como los valores máximos y mínimos que estos soportan.

2.1.13.4. Constantes

Una constante es un nombre significativo que sustituye a un número o una cadena que no varía. Aunque una constante recuerda ligeramente a una variable, no puede modificar una constante o asignarle un valor nuevo como ocurre con una variable.

Declaración de constantes públicas

Declarar una constante pública significa que esa constante podrá ser utilizada desde cualquier procedimiento, formulario o módulo que contenga la aplicación. En la Figura 2.13 observamos un ejemplo de una declaración de una constante:

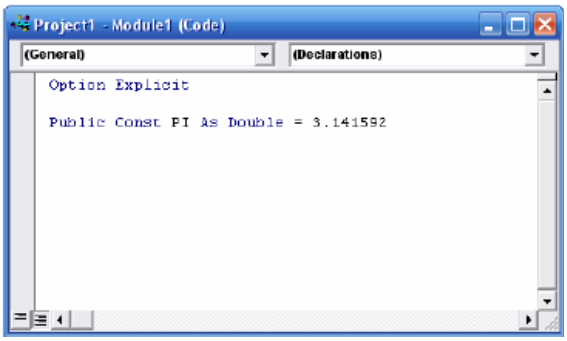


Figura 2.13: Declaración de constantes públicas.

Declaración de constantes privadas

Declarar una constante privada significa que esa constante puede ser usada dentro de todos los procedimientos de un mismo formulario o dentro del formulario donde se declara la constante.

2.1.13.5. Variables

En Visual Basic las variables se utilizan para almacenar temporalmente valores durante la ejecución de la aplicación. Las variables tienen un nombre que nos permite acceder al valor que tiene la variable y un tipo de dato asociado que nos permite determinar la clase de datos que la variable puede almacenar.

Declaración de una variable pública

Al igual que las constantes públicas una variable pública podrá ser utilizada desde cualquier parte de la aplicación. Estas se declaran a nivel de módulos mediante el identificador Public. Una variable pública es declarada mediante el siguiente formato:

Public nombre_variable As Tipo_de_datos

Declaración de una variable privada

Son variables que pueden ser utilizadas dentro de un mismo módulo o formulario. El alcance de este tipo de variable será de nivel general dentro de un mismo módulo o formulario. Se declaran a nivel de módulos o en la sección general de un formulario mediante el identificador Dim. Una variable de nivel privado se declara bajo el siguiente formato:

Dim nombre_variable As Tipo_de_datos

2.1.13.6. Estructuras de control selectivas

Con mucha frecuencia nos encontraremos con la necesidad de hacer comparaciones lógicas y tomar decisiones según los datos proporcionados por el

usuario. Estas decisiones y comparaciones la realizamos utilizando las estructuras de control selectivas.

2.1.13.6.1. Operadores aritméticos

Son aquellos que se utilizan para realizar las operaciones básicas de las matemáticas. En las operaciones básicas tenemos: suma, resta, multiplicación, división, residuo y exponenciación como se puede observar en la Figura 2.14:

Operador	*Significado*	*Ejemplo*	*Resultado*
+	Suma	a + b	Suma de a y b.
-	Resta	a – b	Diferencia entre a y b.
*	Multiplicación	a * b	Producto de a por b.
/	División	a / b	Cociente de a sobre b.
\	División entera	a \ b	Cociente entero de a sobre b.
Mod	Módulo o Resto	a mod b	Resto de a sobre b.
^	Exponenciación	a ^ b	Potencia de a elevado a b.

Figura 2.14: Operadores aritméticos.

2.1.13.6.2. Operadores de relación

Los operadores de relación son utilizados para expresar condiciones y describir una relación entre dos valores. Los operadores de relación se muestran en la siguiente tabla:

Operador	*Significado*	*Equivalente matemático*
>	Mayor que	>
<	Menor que	<
=	Igual a	=
>=	Mayor o igual que	≥
<=	Menor o igual que	≤
<>	Distinto a	≠

Figura 2.15: Operadores de relación.

2.1.13.6.3. La sentencia If

Dado que una condición produce un valor verdadero o falso, se necesita una sentencia de control que ejecute determinada sentencia si la condición es verdadera, y otra si es falsa. En Pascal esta alternativa se realiza con la sentencia

if-then-else. A continuación se describe el diagrama de flujo y el formato de la sentencia.

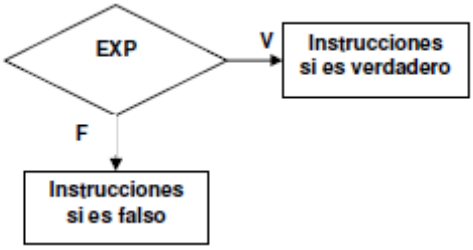


Figura 2.16: Diagrama de flujo de la sentencia if.

La sentencia if es utilizada mediante el siguiente formato:
If (Condición) Then
[instrucciones si es verdadero]
.
Else
[instrucciones si es falso]
.
End If

La condición es una expresión Booleana que puede ser verdadera o falsa (true o false). Una expresión Booleana se forma comparando valores de las expresiones utilizando operadores de relación (relacionales) o comparación y los operadores lógicos vistos anteriormente.

2.2. MIKROBASIC[14]

2.2.1. INTRODUCCIÓN A MIKROBASIC[14]

MikroBasic es una poderosa herramienta de desarrollo para microcontroladores PIC. Está diseñado para proporcionar al cliente la solución más fácil posible para el desarrollo de aplicaciones.

[14] Traducción de la ayuda de Mikrobasic 7.0

2.2.2. IDE INFORMACIÓN GENERAL[14]

MikroBasic es un fácil de utilizar y entorno intuitivo:

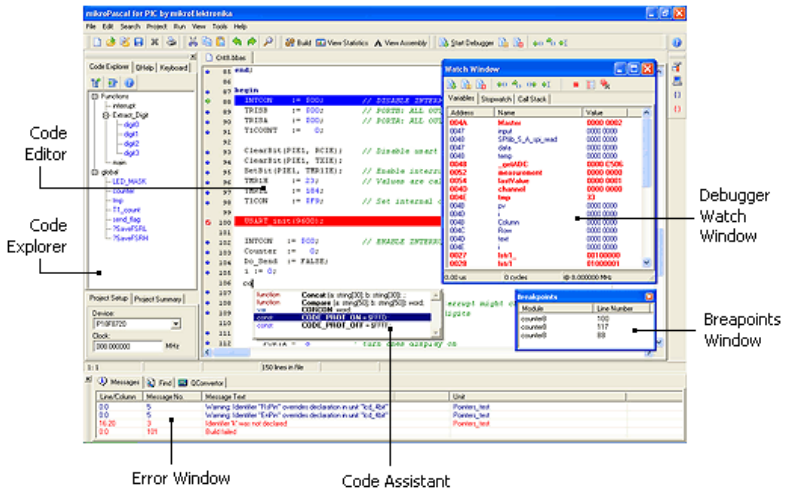


Figura 2.17: Entorno general de Mikrobasic

- El editor de código (Code Editor) es una pantalla que permite escribir el código y corregir errores ortográficos.
- El Explorador de Código (Code Explorer), con métodos abreviados de teclado y un navegador de Ayuda Rápida, está a su disposición para la gestión más fácil de los proyectos.
- La ventana de errores (Error Windows) muestra todos los errores detectados durante la compilación.
- El depurador (Debugger Wath Windows) permite depurar la lógica ejecutable paso a paso al ver el flujo del programa.
- El Asistente para nuevos proyectos (Code Assistant) es una manera rápida, fiable y fácil de crear un proyecto.
- Los archivos de ayuda son la sintaxis y el contexto.
- Como con cualquier aplicación de Windows, es posible personalizar el diseño de MikroBasic que mejor se adapte a sus necesidades.

2.2.3. CARACTERÍSTICAS DE MIKROBASIC[14]

2.2.3.1. Predefiniendo globales y constantes

Para facilitar la programación, MikroBasic implementa un número predefinido de Variables globales y constantes. Todos los Registros SFR del PIC están declarados implícitamente como variables globales del tipo byte, y son visibles en todo el proyecto. Cuando se crea un proyecto, MiKrobasic incluirá el archivo apropiado con extensión .def, el cual contiene las declaraciones disponibles del SFR y constantes (como PORTB, TMR1, etc.). Los identificadores están todos en mayúsculas, idénticos a las nomenclaturas de las hojas de datos de MICROCHIP.

2.2.3.2. Accesando a los bits individuales

Mikrobasic permite acceder de forma individual a cada Bit de una variable. Simplemente use un punto (**.**) con la variable, seguido de un numero.
Por Ejemplo:

```
Dim myvar as longint
' rango de los bits de myvar es de myvar.0...myvar.31
'...
' si RB0 es 1, pon 1 en el 28vo. bit de myvar:
if PORTB.0 = 1 then
myvar.27 = 1
end if
```

No hay necesidad por una declaración en especial; este tipo de acceso selectivo es una característica intrínseca de MikroBasic y puede ser usada en cualquier parte del código. Asegúrese de no exceder el tamaño del tipo apropiado.(P.E. PORTB.12 retornará un error ya que el PORTB es una variable de 8 bits). Si está familiarizado con un Chip en particular, puede acceder a sus bits por su nombre:

```
' Limpiar TMR0F
INTCON.TMR0F = 0
```

Vea Predefiniendo Globales y Constantes para más información de los nombres de REGISTROS/BIT.

2.2.4. ESPECIFICACIONES DEL PIC[14]

Para conseguir el mejor rendimiento del compilador MikroBasic, usted debe estar familiarizado conciertos aspectos del PICmicro. Estos conocimientos no son esenciales, pero pueden ayudar entender mejor las posibilidades y limitaciones del PIC, y su impacto en la escritura del código.

Tipos de Eficiencia

Primero que nada, debe saber que la ALU del PIC, quien realiza las operaciones aritméticas, esta optimizada para trabajar con Bytes. Aunque MikroBasic es capaz de trabajar con tipos de datos muy complejos, el PIC puede chocar con estos, especialmente si esta trabajando con modelos muy viejos. Esto puede aumentar dramáticamente el tiempo necesario para realizar hasta una simple operación. Un consejo universal es usar el tipo más pequeño posible en cada situación. Esto aplica en general para toda la programación, y doblemente para los Microcontroladores.

Conozca su herramienta, Cuando haga cálculos, no todos los PIC tienen el mismo rendimiento. Por ejemplo, la Familia PIC16 no tiene Hardware para multiplicar 2 bytes, pero se compensa con un algoritmo por software. Por otro lado, la familia PIC18 tiene un hardware multiplicador, y por ende la multiplicación trabaja considerablemente más rápido

Limitaciones de Llamadas Anidadas.

Las llamadas anidadas representan una llamada a una función dentro del cuerpo de la función. O así mismo (llamadas recursivas) a otra función. Las llamadas recursivas, son una forma de cruzar las llamadas, y no están soportadas por MikroBasic debido a la pila del PIC y limitaciones de memoria.

MikroBasic limita el número de llamadas anidadas no recursivas a:

- 8 llamadas para la familia PIC12
- 8 llamadas para la familia PIC16
- 31 llamadas para la familia PIC18

2.2.5. ELEMENTOS LÉXICOS[14]

Estos temas proporcionan una definición formal de los elementos léxicos de MikroBasic. Ellos describen las diferentes categorías de unidad de tipo de palabra (TOKENS) reconocidas por el lenguaje.

2.2.5.1. Espacios en Blanco

Los espacios en blanco es el nombre dado a los espacios (blancos), horizontales o verticales y comentarios. Los espacios en blanco sirven para indicar donde empiezan los Tokens y donde terminan. Mas allá de esto cualquier espacio en blanco es descartado.

Por ejemplo:
dim tmp **as byte**
dim j **as word**

2.2.5.2. Espacios en Blanco en cadenas

Los caracteres ASCII representando un espacio en blanco pueden darse dentro de las cadenas literales, en cuyo caso están protegidos por el análisis gramatical normal (permanecen como parte de la cadena).Por ejemplo: la declaración
some_string = "mikro foo"

2.2.5.3. Comentarios

Los comentarios son pedazos de texto usados para hacer anotaciones en el programa, y son técnicamente otra forma de espacios en blanco. Use el apostrofe para crear un comentario:
' Cualquier texto en medio de un apostrofe y el fin de la línea

2.2.6. TOKENS[14]

Un Token es el elemento mas pequeño en BASIC que es significativo para el compilador, El analizador separa los Tokens de la entrada creando un Token mas largo usando los caracteres de izquierda a derecha.
MikroBasic reconoce los siguientes tipos de Tokens:

- Palabras Clave
- Identificadores
- Constantes
- Operadores
- Signos de puntuación (también conocidos como Separadores)

2.2.6.1. Literales

Las Literales son tokens que representan un valor numérico o el valor de un carácter. El tipo de dato de una constante es deducido por el compilador, usando pistas como el valor numérico y el formato usado en el código fuente.

2.2.6.2. Enteros

Los valores enteros pueden ser representados en formato Decimal, Hexadecimal o en notación binaria. En notación Decimal, los numerales son representados como una secuencia de dígitos (sin comas, espacios, o puntos), con los prefijos opcionales + o para indicar el signo. Por default los valores son positivos (6258 es equivalente a +6258).

El prefijo signo de pesos o dólar ($) o el prefijo 0x indica un numero hexadecimal (por ejemplo, $8F o 0x8F). El prefijo signo de porcentaje (%) indica un número binario (por ejemplo %0101). Aquí hay algunos ejemplos:

```
11      ' Valor decimal
$11     ' Valor Hexadecimal, Igual a 17 decimal
0x11    ' Valor Hexadecimal, Igual a 17 decimal
%11     ' Valor Binario, Igual a 3 decimal
```

El rango máximo permitido de un valor es impuesto por la longitud del tipo de dato en MikroBasic longint. El compilador reportará un error si la literal excede el valor 2147483647 ($7FFFFFFF).

2.2.6.3. Punto flotante

Un valor de punto flotante consiste en:

- Un entero decimal
- El punto decimal
- La fracción decimal

MikroBasic limita las constantes de punto flotante a un rango de +1.17549435082* 10e18...+ 6.80564774407* 10e38. Aquí algunos ejemplos:

```
' = 1.23
23.45e6        ' = 23.45 * 10^6
2e5
' = 2.0 * 10^5
3E+10          ' = 3.0 * 10^10
.09E34         ' = 0.09 * 10^34
```

2.2.6.4. Caracteres

Las literales de caracteres son solamente un carácter del código ASCCI, dentro de comillas (Por ejemplo, "A"). El carácter puede ser asignado a variables de tipo

Byte y Char (la variable byte asignará el valor ASCII del carácter). también puede asignar variables carácter a variables tipo cadena.

2.2.6.5. Cadenas

Una cadena es una secuencia de 255 caracteres del código ASCII, dentro de comillas. Los espacios en blanco son preservados dentro de las cadenas. Por ejemplo el analizador ve la siguiente cadena: "IR A" como un simple token.

La longitud de las cadenas depende del número del que consisten. Las cadenas se guardan internamente como la sucesión dada de caracteres más un carácter nulo final (el cero del ASCII). Las cadenas que no tengan nada dentro de las comillas son guardadas como un carácter nulo. Puede asignar cadenas a variables de cadena o a un arreglo de variables char.

Algunos ejemplos:
"Hello world!" ' mensaje, 12 caracteres de longitud
"Temperature is stable" ' mensaje, 21 caracteres de longitud

2.2.6.6. Palabras Clave

Las palabras clave son palabras reservadas para propósitos especiales y no pueden ser usadas como un nombre de un identificador.
Las palabras clave de Basic y las relevantes del registro SFR son definidas como variables globales y representan palabras reservadas que no pueden ser redefinidas como por ejemplo:
PORTB, TIMER0, T1CON, etc.

Aquí está el listado alfabético de algunas palabras reservadas en MikroBasic:

- absolute
- abs
- and

- array
- asm
- begin
- boolean
- case
- char
- chr
- clear
- const
- dim
- div
- do
- double
- for
- function
- goto
- gosub
- if
- include
- in
- int
- is
- loop
- label
- mod
- module
- new
- next
- switch
- then
- to
- while

2.2.7. ORGANIZACIÓN DEL PROGRAMA[14]

2.2.7.1. Organización del Módulo Principal

Básicamente, el código fuente tiene dos secciones: declaraciones y el cuerpo del programa. Las declaraciones deben ir en la parte apropiada del código, organizadas de manera ordenada, de otro forma el compilador no será capaz de comprender el programa correctamente.

Cuando escriba el código, siga el modelo que se presenta debajo. El módulo principal deberá lucir de la siguiente manera:

program <nombre del programa>
include <incluir otros módulos>
'***
'* Declaraciones Globales:
'***
' Declaraciones simbólicas
symbol ...
' declaración de constantes
const ...
'declaración de Variables
dim ...
' declaración de procedimientos
sub procedure nombre del procedimiento(...)
<Declaraciones locales>
...
end sub
' declaración de funciones
sub function nombre de la función(...)
<declaraciones globales>
...
end sub

```
'*******************************************************
'* Cuerpo del Programa
'*******************************************************
main:
' Escriba el código aqui
end.
```

2.2.8. MÓDULO PRINCIPAL (MAIN) [14]

Cada proyecto en MikroBasic requiere de un único módulo principal (main) el módulo principal es identificado por la palabra reservada program al inicio; esto sirve para que el compilador sepa por dónde empezar.

Después de que haya creado un nuevo proyecto con el asistente de proyecto, el editor de código desplegará un nuevo módulo principal, el cual contiene partes de un programa.

```
program MyProject
' main procedure
main:
' Place program code here
end.
```

Nada que no sea un comentario debe ir precedido por la palabra reservada program después del nombre del programa usted puede incluir opcionalmente las cláusulas include. Ponga todas las declaraciones globales (constantes, variables, etiquetas, rutinas) antes de la etiqueta main.

En mikrobasic, la declaración end. (la declaración de finalización de cualquier programa) actúa como un ciclo infinito.

2.2.9. VARIABLES[14]

Una variable es un objeto cuyo valor puede ser cambiado durante el tiempo de ejecución. Cada variable debe ser declarada bajo un único nombre de identificador válido. Este nombre es usado para acceder al lugar de la memoria que ocupa la variable.

Cada variable es declarada en la sección de declaración de cualquier archivo o rutina, cada variable debe ser declarada antes de ser usada. Las variables Globales (aquellas que no están encerradas por ninguna rutina o procedimiento) son declaradas debajo de las cláusulas include y arriba de la etiqueta main.

Es imperativo especificar el tipo de variables en MikroBasic. La sintaxis para declara una variable en MikroBasic es:

dim nombre_de_identificador **as** type

Donde nombre del identificador puede estar delimitado por una coma para separar diferentes nombres de identificadores y type es el tipo de dato soportado por MikroBasic.

2.2.10. VARIABLES Y EL PIC[14]

Cada variable declarada consume una parte de la memoria RAM del PIC. El tipo de dato de la variable no solo delimita el rango de la variable sino también determina el espacio usado en la memoria RAM del PIC. Tenga en mente que realizar operaciones con diferentes tipos de variables toma diferentes tiempos en ser completadas. MikroBasic recicla el espacio de variables locales – las variables locales declaradas en diferentes rutinas y procedimientos comparten el mismo espacio de memoria si es posible.

2.2.11. CONSTANTES[14]

Una constante es un dato cuyo valor no puede ser cambiado durante el tiempo de ejecución. Usar una constante en un programa no consume memoria RAM de un PIC. Una constante puede usarse en cualquier expresión pero no se le puede asignar otro valor.
Las constantes son declaradas en la sección de declaración de un programa. La sintaxis de declaración de una constante deberá ser así:

const constant_name [**as** type] = value

Cada constante es declarada bajo un único nombre, el cual debe ser un identificador válido. Es una tradición escribir las constantes en mayúsculas. Una constante necesita que se especifique su valor (value) que es una literal apropiada para el tipo de dato usado. El tipo (type) es opcional; en la ausencia del tipo el compilador asume el tipo de dato mas pequeño el cual se ajuste al tamaño del valor. Aquí algunos ejemplos:

const MAX **as longint** = 10000
const MIN = 1000 ' El compilador asume el tipo de dato como word
const SWITCH = "n" ' El compilador asume el tipo de dato como char
const MSG = "Hello" ' El compilador asume el tipo de dato com[14]o cadena
const MONTHS **as byte**[12] = (31,28,31,30,31,30,31,31,30,31,30,31)

2.2.12. DECLARACIONES[14]

Las declaraciones definen la manera en que actúan los algoritmos dentro de un programa. Cada declaración necesita estar determinada por una nueva línea de carácter (enter).

Esto puede ser usado para crear ciclos, saltos, y otras declaraciones estructuradas. En la ausencia de saltos y selección de declaraciones, las

declaraciones serán ejecutadas de manera secuencial tal como estan escritas en el código fuente. Véase:

- Declaraciones de asignación
- Declaraciones condicionales
- Declaraciones de iteración (ciclos)
- Declaraciones de saltos
- Declaraciones de lenguaje Ensamblador (ASM).

2.2.12.1. Declaraciones de asignación

Las declaraciones de asignación tiene la siguiente forma:
variable = expresión

La declaración evalúa la expresión y le asigna el valor de la variable. Todas las reglas de la conversión implícita se aplican. Variable puede ser declarada como una variable o como un arreglo y expresión puede ser cualquier expresión. No confunda el asignamiento con el operador relacional igual (=), que prueba una igualdad. MikroBasic interpretará el signo igual (=) dentro de su contexto.

2.2.12.2. Declaraciones condicionales

Las declaraciones condicionales o de selección, escogen diferentes tipos de acciones dependiendo de los valores o expresiones evaluadas.

Declaraciones IF
Use declaraciones if para implementar una declaración condicional. La sintaxis de las declaraciones IF es la siguiente:

if expresion **then**
declaraciones
[**else**
Otras declaraciones]
end if

Donde la expresión, es evaluada si es verdadero las declaraciones se ejecutan. Si la expresión es falsa se ejecutaran otras declaraciones. La expresión debe ser convertida a tipo boolean, de otra forma la expresión esta mal formulada. La palabra clave else y las otras declaraciones son opcionales.

Declaraciones de IF's anidados

Anidar If requiere de atención especial, la regla general es que la condición anidada se analiza desde la condición mas profunda, con cada else limitando al if más cercano de la izquierda.

2.2.12.3. Declaraciones de iteración.

Las declaraciones de iteración le permiten crear ciclos de un set de declaraciones.

Declaraciones FOR

La declaración de un FOR determina un ciclo de iteración y requiere que se especifique el número de iteraciones. La sintaxis de una sentencia For es:

for contador = valor inicial **to** valor final [**step** valor de paso]
declaraciones
next contador

El contador es una variable que se incrementa según el valor del paso con cada iteración del ciclo. El parámetro valor de paso es un valor entero opcional y si este es omitido el valor por defecto es 1, antes de la primera iteración, el valor del contador es puesto al valor inicial y se incrementa mientras no exceda o no llegue al valor final, con cada iteración las declaraciones son ejecutadas.

Los valores iniciales y finales deben ser valores compatibles con el contador; las declaraciones pueden ser cualquier tipo de declaraciones siempre y cuando no afecten el valor del contador.

Aquí un ejemplo de cómo calcular el producto escalar de 2 vectores a y b, de longitud n usando la sentencia FOR:

```
S = 0
for i = 0 to n
s = s + a[i] * b[i]
next i
```

Ciclo infinito

La sentencia for puede resultar un ciclo infinito si el valor final es igual o excede el rango del tipo de dato del contador. Por ejemplo, esto resultara un ciclo infinito ya que el contador nunca alcanzara el valor 300:

```
dim counter as byte
...
for counter = 0 to 300
nop
next counter
```

CAPÍTULO III:
DISEÑO Y ENSAMBLAJE

3.1. DISEÑO Y CONSTRUCCIÓN DEL HARDWARE

3.1.1. DESCRIPCIÓN GENERAL DEL PROYECTO

Este prototipo educativo simula un sistema de vigilancia y seguridad en una casa, el cual va a ser monitoreado por una persona que conozca sobre el uso del mismo y que funciona con un PIC16F877A, a los pines de este PIC están conectados ocho luminarias, un sensor de presencia, un sensor infrarrojo, un sensor de humo y dos sensores magnéticos, además la programación del PIC se lo realizó en MikroBasic 7.0.

La vigilancia de la casa se lo realiza a través de un programa diseñado en Visual Basic 6.0 que nos permite monitorear físicamente lo que sucede en nuestra casa, esto quiere decir que si se activan los sensores de presencia, humo, magnéticos o se enciende o apaga una luminaria, el programa nos indica en que parte de la casa se está activando.

Este programa consta de un teclado el mismo que nos permite activar la alarma para que después de 5 segundos comience a funcionar los diferentes sensores, cuando exista un intruso en la casa inmediatamente la alarma sonará a través de un buzzer, la misma que podemos desactivar si existió una falsa alarma de presencia, además este teclado nos permite cambiar de clave.

El control de luces lo podemos hacer manualmente por medio de cualquiera de los 8 interruptores o también podemos hacerlo a través del programa de control de Visual Basic, de tal forma que los interruptores funcionan como un conmutador.

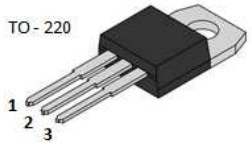
En la planta baja de la casa podemos encontrar: 2 sensores magnéticos que están conectados a la ventana y la puerta respectivamente, y un sensor de presencia ubicado en la sala, los mismos que son utilizados para advertir la presencia de algún intruso. El sensor infrarrojo está ubicado en las gradas, cuando detecta presencia automáticamente el foco de las gradas se enciende caso contrario se apaga. Además en la cocina podemos encontrar un sensor de humo que al detectar la presencia de humo en el aire va a emitir una señal acústica por medio de un buzzer que le hemos incorporado.

Se ha programado el encendido y apagado de las luces externas de la casa para controlar los horarios de uso, por lo tanto se pueden programar la cantidad de tiempo que deberán estar encendidas las luces externas.

3.1.2. ETAPA DE ALIMENTACIÓN

Consta de una fuente de poder de 24V y 500mA la cual es alimentada por 110V y 60Hz, también consta de un regulador de voltaje de 5V fijos y 250mA (7805), y un regulador de voltaje de 12V y 250mA (7812), y con la ayuda de los condensadores conectados a estos reguladores obtendremos una señal continua pura de 5V y 12V respectivamente.

El tipo de encapsulado que usamos para los reguladores 7805 y 7812 es el TO-220 ver Figura 3.1, estos reguladores tiene protecciones por sobrecarga térmica, y contra cortocircuitos.



1 Input, 2 GND, 3 Output

Figura 3.1: Descripcion de pines.

El 7805 permite un voltaje de entrada de hasta 35V, un voltaje de salida de 5V y soporta una temperatura de 120℃.

Mientras que el 7812 permite un voltaje de entrada de hasta 40V, a la salida obtendremos un voltaje de 12V y esta diseñado para soportar una temperatura de 120℃.

En la Figura 3.2 se muestra como se configuró el 7805 y el 7812 para el presente proyecto.

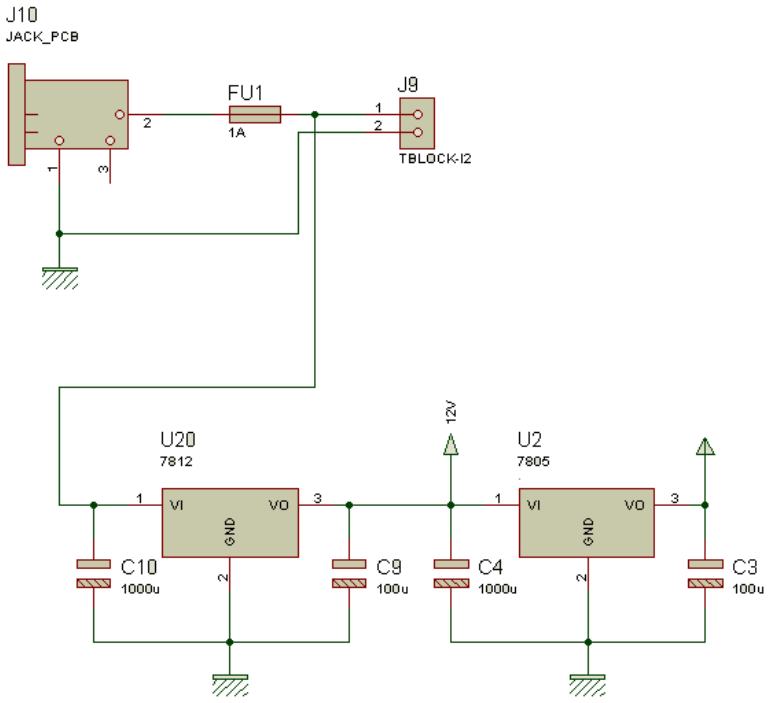


Figura 3.2: Conexión del 7805 y 7812.

En la Figura 3.3 se realizó la distribucion de voltaje que usamos en cada etapa del prototipo para el correcto funcionamieto del mismo.

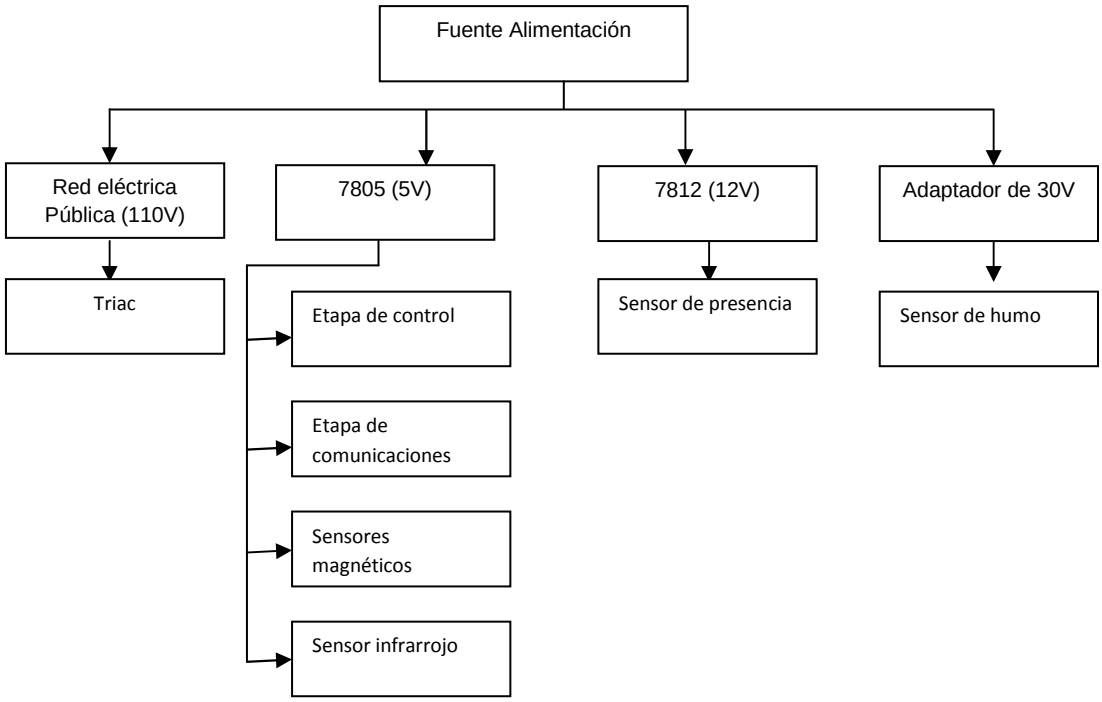


Figura 3.3: Distribucion de voltajes para cada etapa del prototipo.

3.1.3. ETAPA DE CONTROL

El microcontrolador es el encargado de tomar decisiones de activar y desactivar la alarma, prender y apagar los focos, y enviar y transmitir los datos a la PC en donde se visualizara mediante un programa diseñado en Visual Basic en forma gráfica que sensor fue activado, también tendrá la opción de programar a una determinada hora para que se prendan y apaguen las luces que se encuentran en la parte exterior de nuestro prototipo.

Utilizamos para este proyecto el microcontrolador PIC16F877A ya que tiene las características que requeríamos para la realización del prototipo, entre ellas:

- Hasta 8192 palabras de memoria Flash
- Hasta 368 bytes de memoria de datos (SRAM)
- Hasta 256 bytes de memoria de datos EEPROM
- 33 pines de entrada/salida de los cuales usamos 22 pines
- Comunicación vía usart (Receptor/Transmisor Sincrónico/Asincrónico Universal)

A continuación vamos a describir cada uno de los pines que pertenecen a la etapa de control:

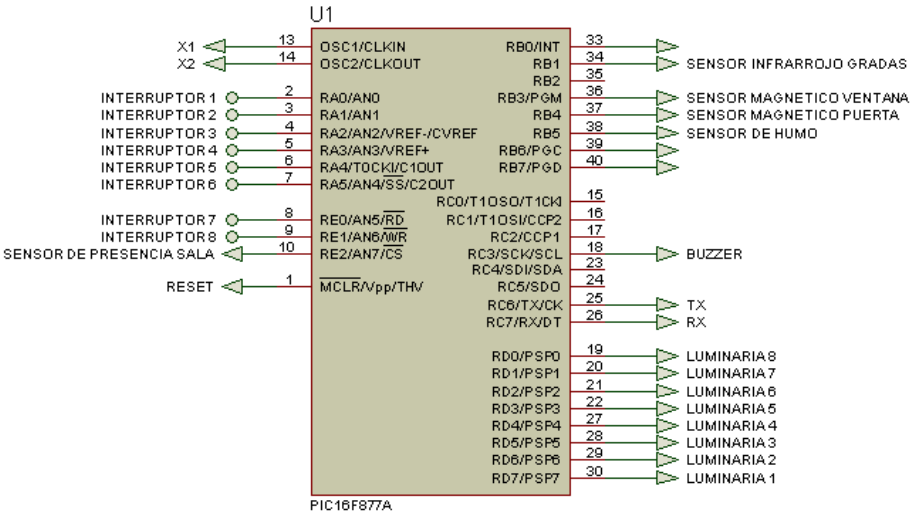


Figura 3.4: Descripción de pines del PIC 16F877A.

3.1.3.1. Pin 1

Este pin es llamado $\overline{\text{MCLR}}$, cuando esté en 0 lógico nos permite dar un RESET general al PIC, y cuando este en 1 lógico el PIC funcionará normalmente con el programa que este previamente cargado. Además se debe conectar un pulsador y una resistencia de protección de 1KΩ (R9) que va conectada a Vcc (+5V).

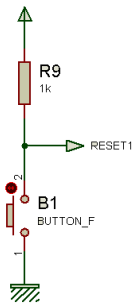


Figura 3.5: Configuración del **$\overline{\text{MCLR}}$.**

3.1.3.2. Pin 2, 3, 4, 5, 6, 7

Son terminales bidireccionales que pueden ser usados como entradas o salidas de datos, estos pertenecen al puerto A (RA0, RA1, RA2, RA3, RA4, RA5).

En el actual proyecto determinamos como entradas, los interruptores que serán el mando físico para el encendido y apagado de las luminarias.

3.1.3.3. Pin 8, 9

Son terminales bidireccionales que pueden ser usados como entradas o salidas de datos, estos pertenecen al puerto E (RE0, RE1). En el actual proyecto determinamos como entradas, los interruptores que serán el mando físico para el encendido y apagado de las luminarias.

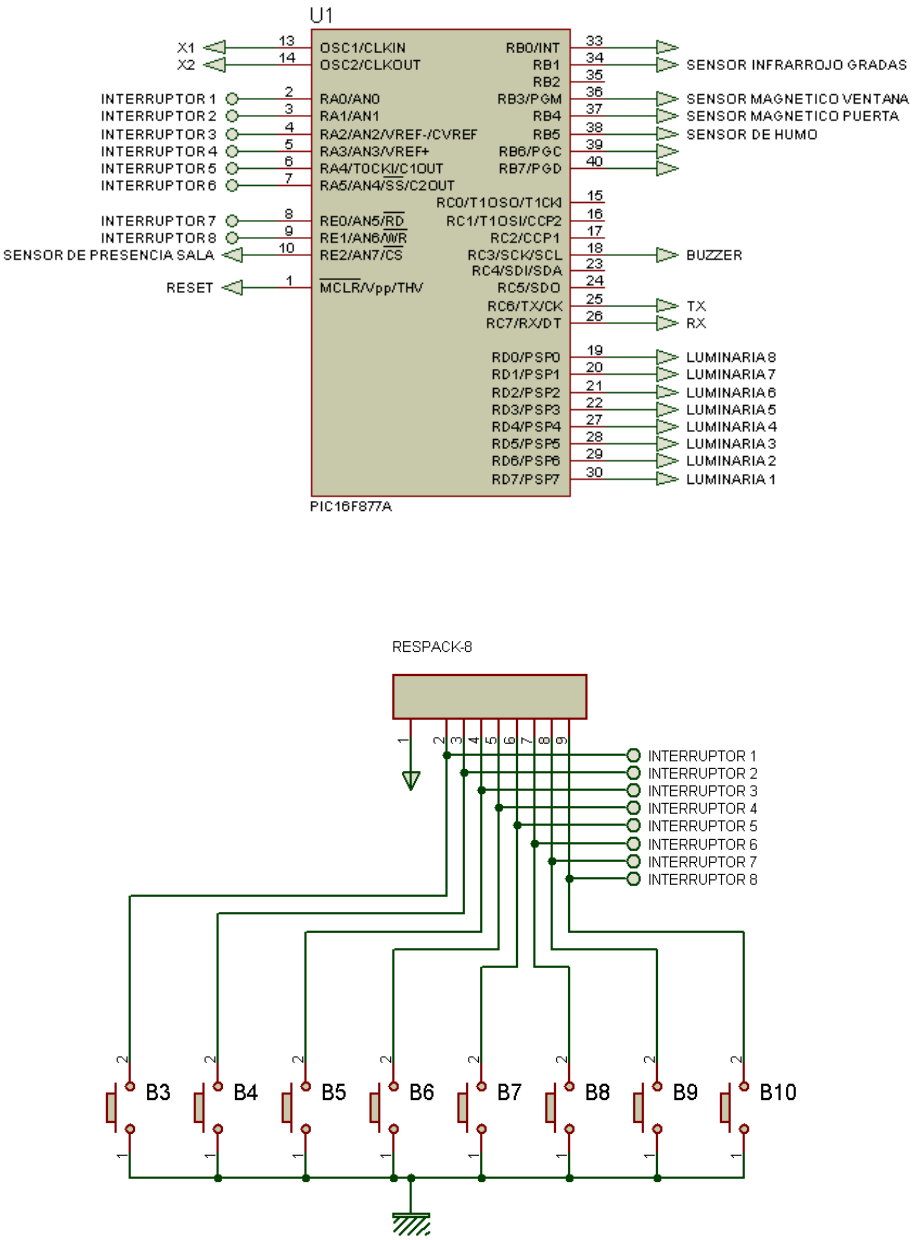


Figura 3.6: Configuración de los pines 2, 3, 4, 5, 6, 7, 8, 9.

3.1.3.4. Pin 11, 32

Es para polarización Vdd ésta va conectada a 5V.

3.1.3.5. Pin 12, 31

Es para polarización Vss va conectado a tierra es decir 0V.

3.1.3.6. Pin 13, 14

El pin 13 es un OSC1/CLKIN, que quiere decir entrada de reloj y el pin 14 es un OCS2/CLKON, que quiere decir salida de reloj. La frecuencia de trabajo del microcontrolador viene dado por el oscilador externo, en este proyecto utilizamos un cristal de 8 MHz, este va conectado a dos condensadores de 22 pF como se muestra en la Figura 3.7.

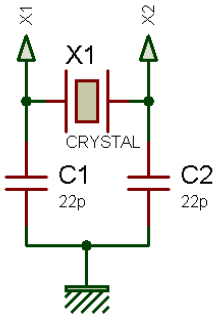


Figura 3.7: Configuración del oscilador externo.

Fórmula para calcular un ciclo de máquina

1 instrucción= 1CM=4T=4/f

1 instrucción= Tiempo que tarda una instrucción en ejecutarse.

1 CM= Un ciclo de máquina.

T= Periodo

f= frecuencia del cristal.

Cálculos:

Datos

f= 8MHz

1 instrucción= 1CM=4T=4/f

1 instrucción= 1CM=4T=4/8MHz

1 instrucción= 1CM= 0,5µseg.

3.1.3.7. Pin 18

Es un terminal bidireccional que pueden ser usado como entrada o salida de datos, en este caso será utilizado como salida y envía 1L en respuesta de la activación de cualquier sensor por lo tanto se activará el buzzer.

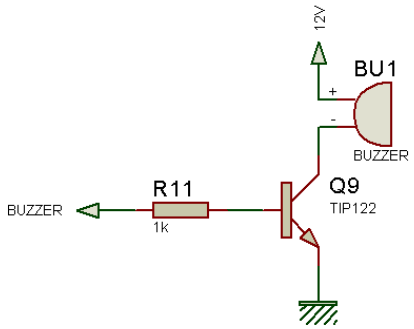


Figura 3.8: Configuración del pin 18.

3.1.3.8. Pin 19, 20, 21, 22, 27, 28, 29, 30

Pertenecen al puerto D son líneas bidireccionales y pueden ser utilizadas como entradas y salidas de datos:
Se determina estos pines como salidas para controlar 8 luminarias.

RD0 (Pin 19): Luminaria 8 (Exterior de la casa).
RD1 (Pin 20): Luminaria 7 (Exterior de la casa).
RD2 (Pin 21): Luminaria 6 (Dormitorio máster).
RD3 (Pin 22): Luminaria 5 (Sala de estar).

RD4 (Pin 27): Luminaria 4 (Dormitorio).
RD5 (Pin 28): Luminaria 3 (Gradas).
RD6 (Pin 29): Luminaria 2 (Comedor).
RD7 (Pin 30): Luminaria 1 (Sala).

En la Figura 3.9 describimos la configuración de una luminaria, la misma que la conectamos al pin RD7, esta configuración es la misma para las siete luminarias restantes.

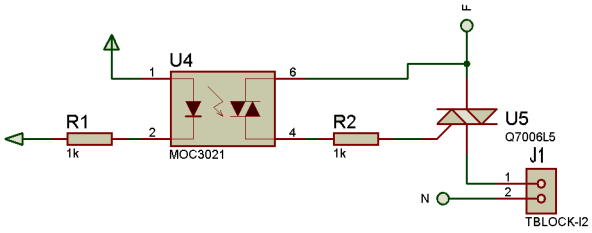


Figura 3.9: Configuración de una luminaria en el puerto D.

3.1.3.9. Pin 25, 26

Son terminales bidireccionales que pueden ser usados como entradas o salidas de datos, estos pertenecen al puerto C (RC6, RC7.), En nuestro proyecto serán utilizados como transmisión y recepción de datos, para la comunicación entre el microcontrolador y la PC por medio del Max 232.

3.1.3.10. Pin 33, 34, 36, 37, 38.

Pertenecen al puerto B son líneas bidireccionales y pueden ser utilizadas como entradas y salidas de datos.

Estos pines son utilizados como entradas, asignados de la siguiente forma:

RB0 (Pin 33): Sensor de presencia (Sala).
RB1 (Pin 34): Sensor infrarrojo (Gradas).
RB2 (Pin 36): Sensor magnético (Ventana).
RB3 (Pin 37): Sensor magnético (Puerta).

RB5 (Pin 38): Sensor de Humo.

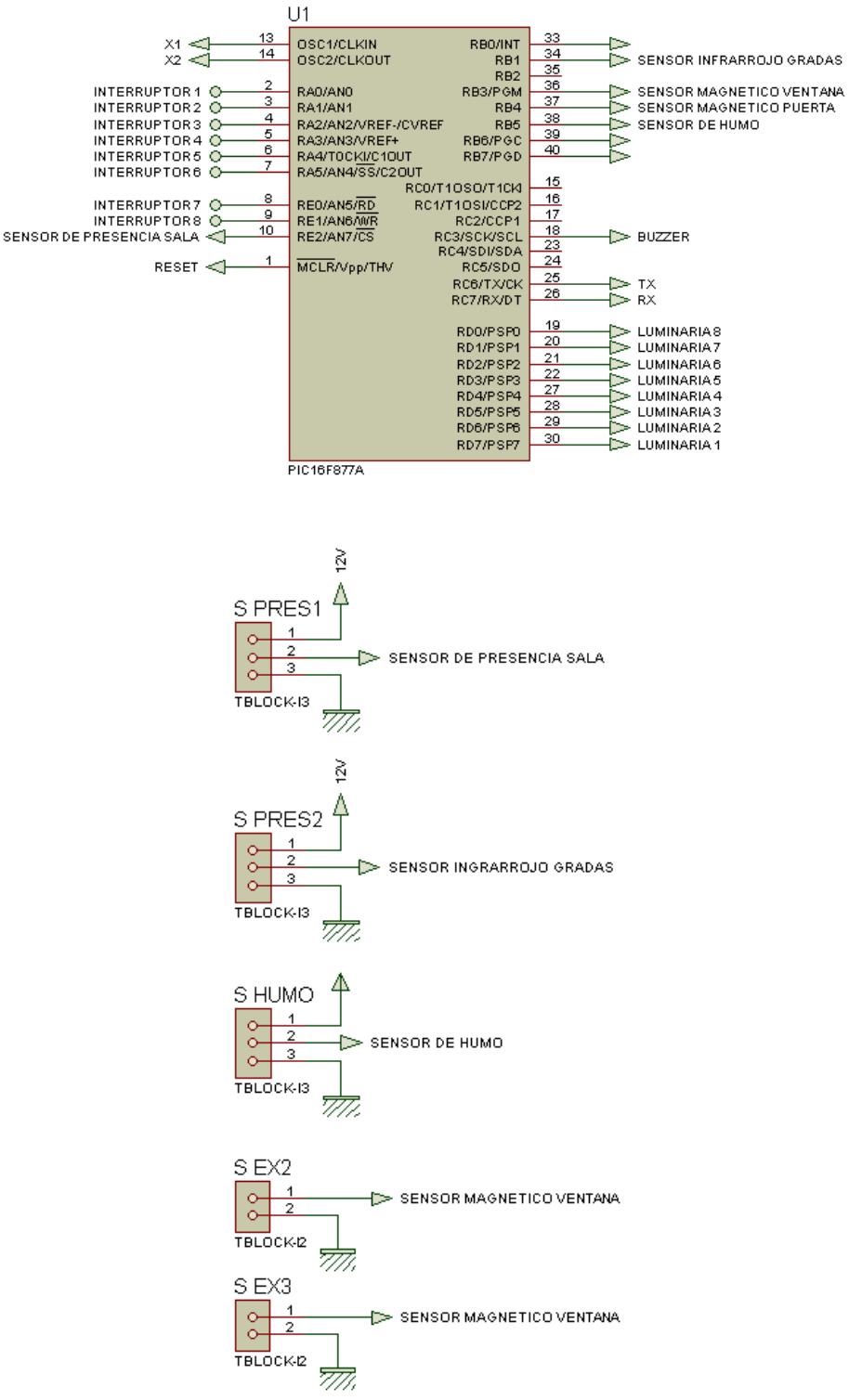


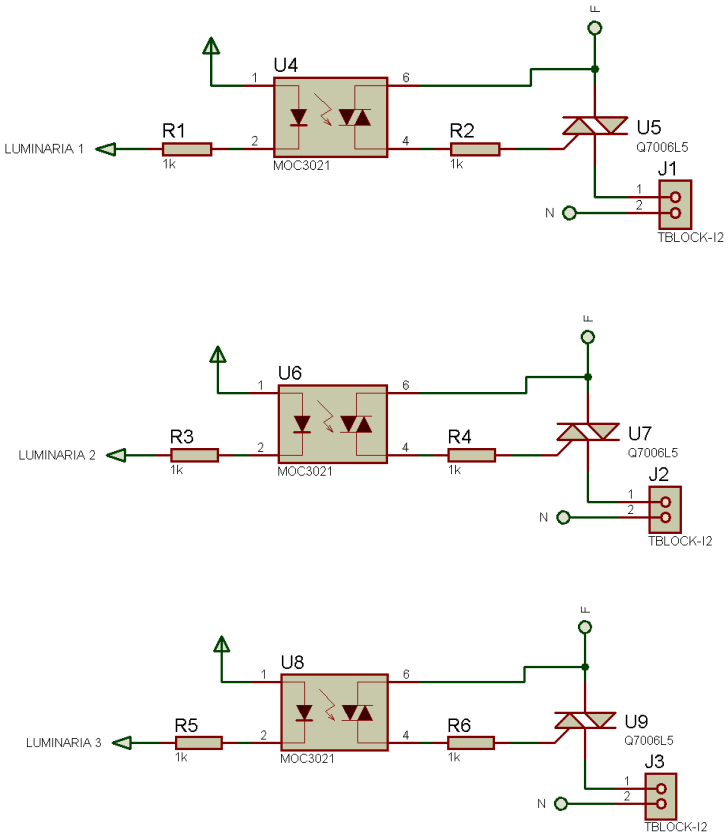
Figura 3.10: Configuración de los pines 33, 34, 36, 37, 38.

3.1.4. ETAPA DE ACTUACIÓN (POTENCIA)

Se utilizaron 8 triacs uno por cada foco (BT139) de 16 A con sus respectivos optoacopladores (Optotriac MOC 3020), los triacs son los dispositivos que permiten manejar corriente alterna como es en este caso. En la Figura 3.11 se puede observar la configuración de cada una de las salidas.

Cada optotriac recibe la señal de las correspondientes salidas del microcontrolador, estas salidas están en 0L o 1L, dependiendo de los valores que sean enviados por el software, y sean interpretados por el microcontrolador.

El optotriac tiene una alimentación TTL de 0 – 5V, para su funcionamiento por lo cual este manda la señal enviada del microcontrolador a la compuerta del triac para que este active a su carga conectada a 110V.



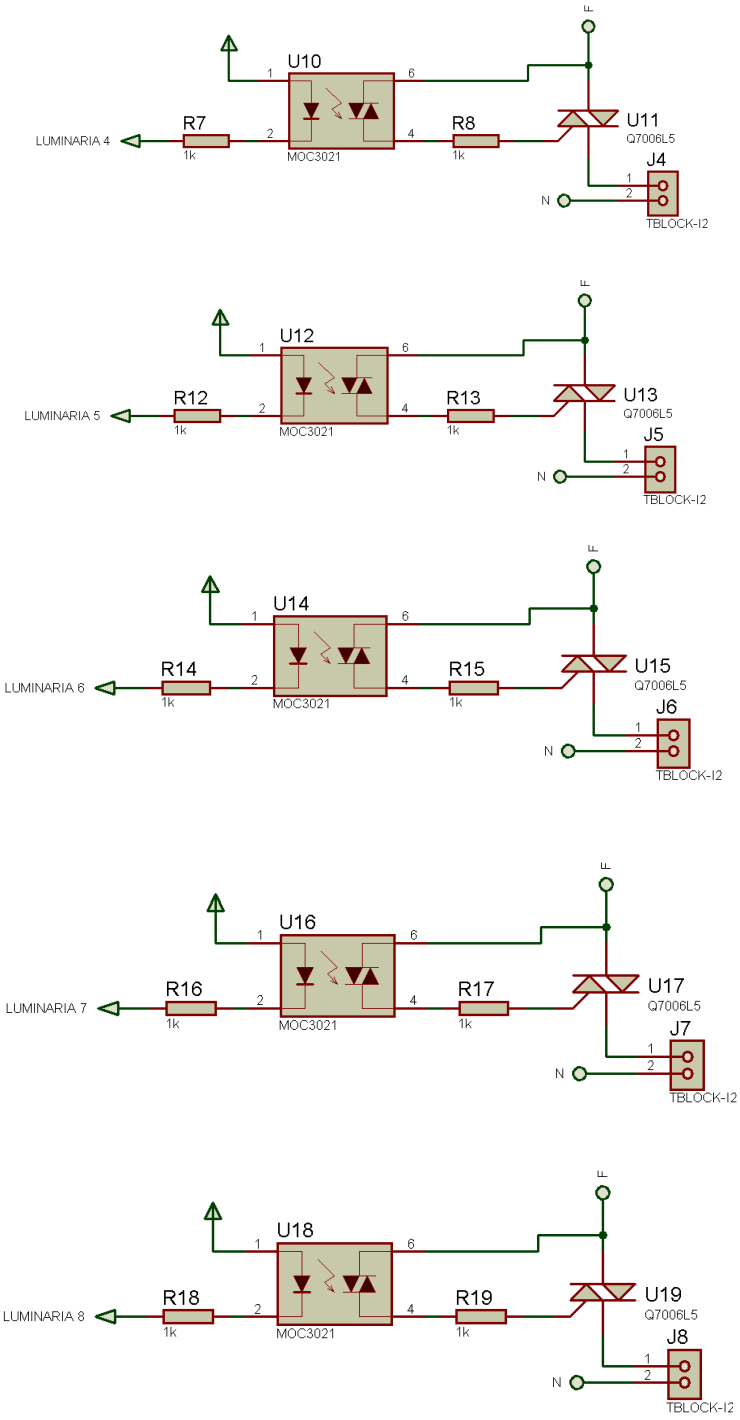


Figura 3.11: Etapa de actuación (potencia).

3.1.4.1. Control On/Off de las luminarias

Control On/Off se controla por medio del software y por medio de forma física que son los interruptores.

3.1.4.2. Control de Buzzer

Este va a ser encendido cuando los sensores magnéticos, de movimiento, infrarrojo sean activados siempre y cuando esté activado el sistema de alarma.

3.1.5. ETAPA DE COMUNICACIÓN

Para la comunicación entre el microcontrolador y la PC, se utilizó el circuito integrado MAX232, el cual es un conversor de señales TTL que salen del microcontrolador a señales que utiliza la interfaz RS-232 y viceversa.

En la Figura 3.12 se muestra la configuración del MAX232.

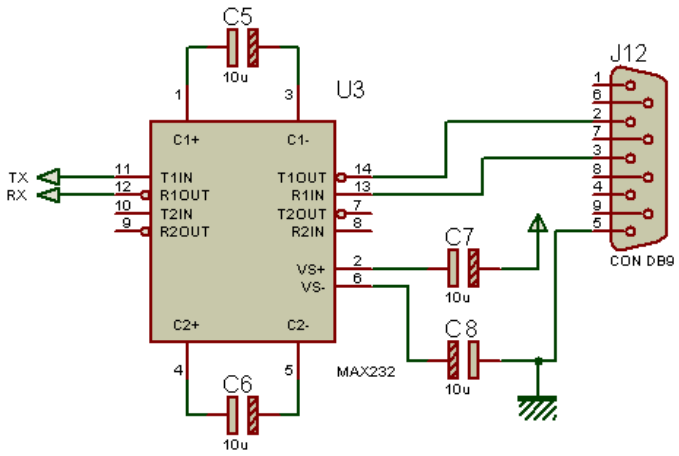


Figura 3.12: Configuracion del MAX232.

3.1.5.1. Transmision de datos

Esta etapa de transmisión y recepción de datos lo realizamos con la utilización del computador, el MAX232 y el circuito de control. Esta es la encargada de entregar una señal digital. Esta comunicación se realiza mediante comunicación serial utilizando el puerto de comunicación serial, con un conector DB9, mediante los pin 2 donde ingresa los datos (recepción) y pin 3 donde envian los datos (transmición), conectados al circuito MAX232 en los pines 13 y 14.

3.1.6. ETAPA DE SOFTWARE

3.1.6.1. Diagramas de flujo

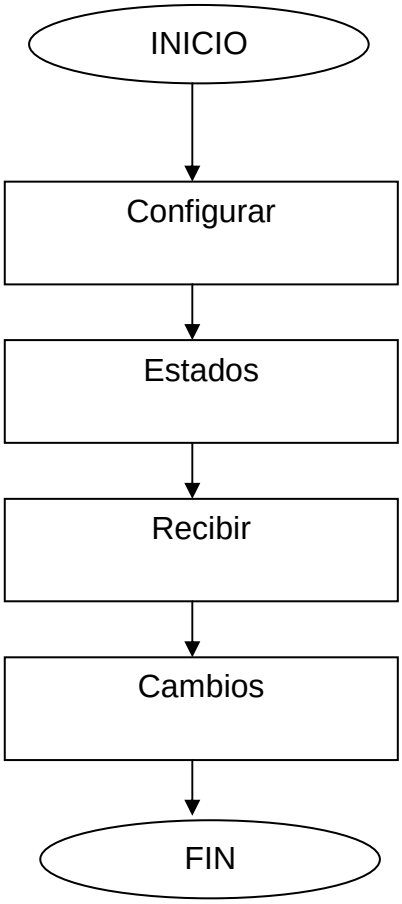
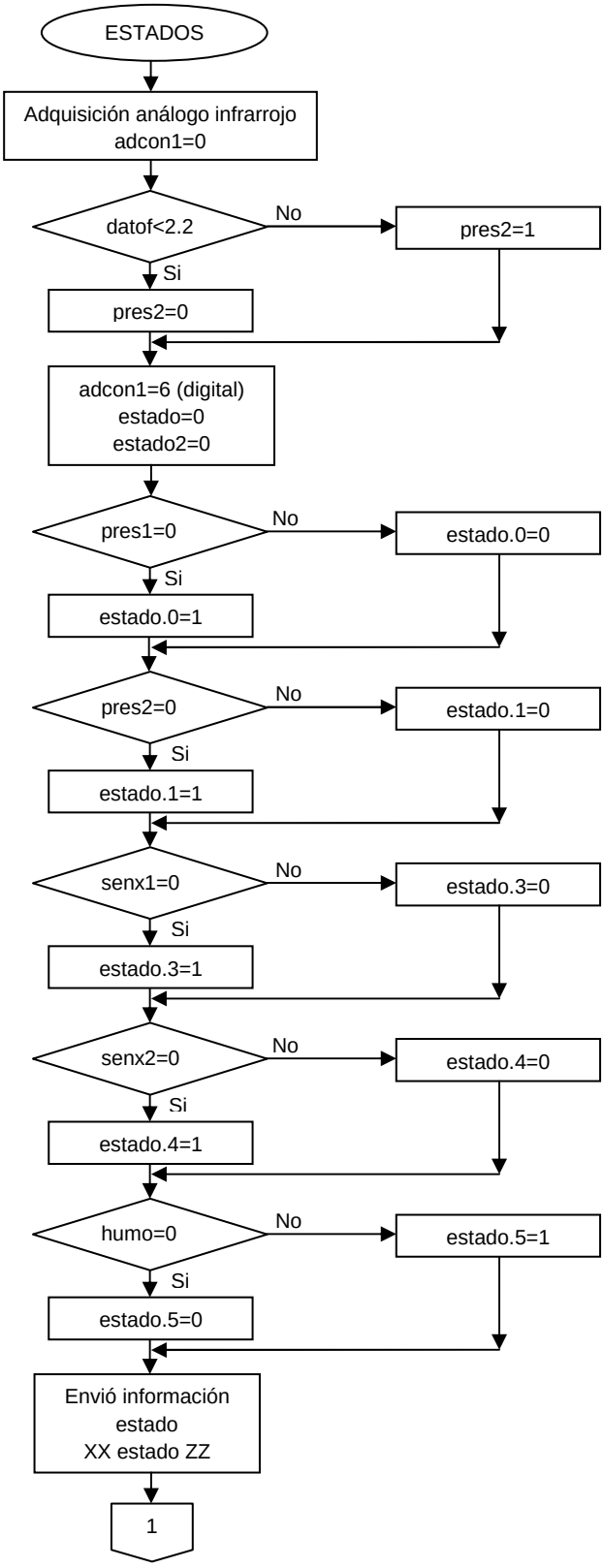
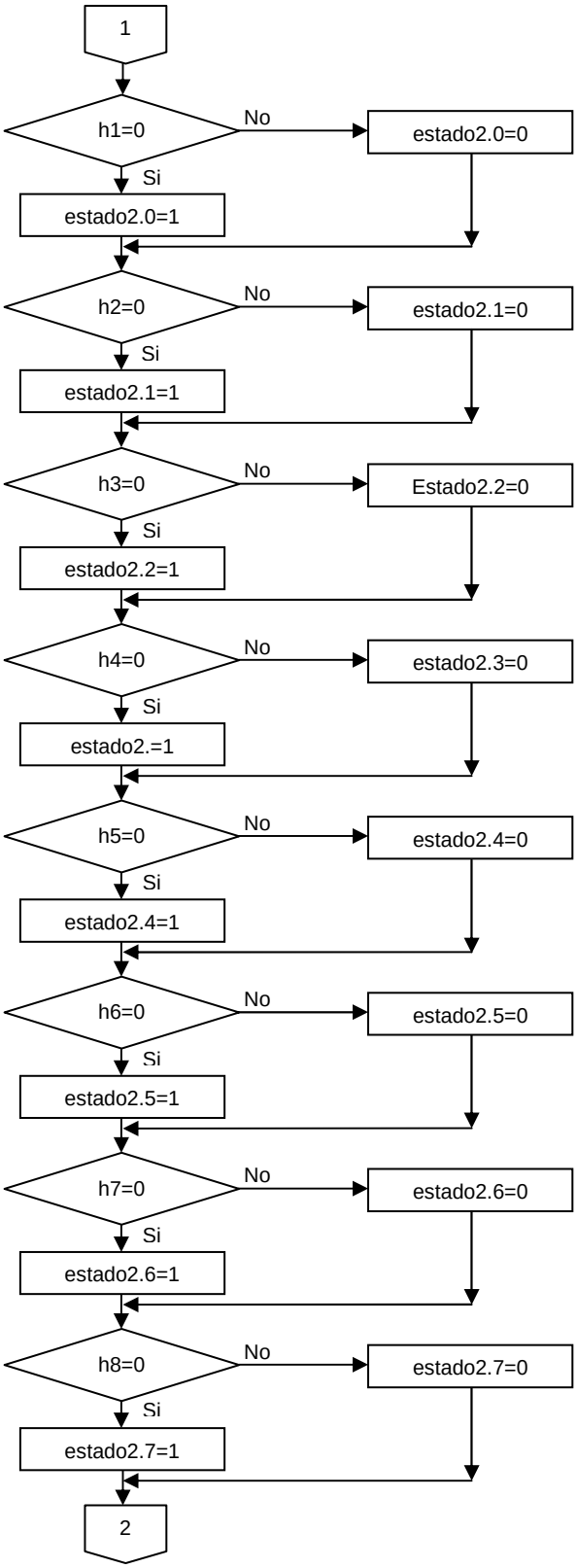


Figura 3.13: Diagrama de flujo general.


ESTADOS
Adquisición análogo infrarrojo
adcon1=0
datof<2.2
No
pres2=1
Si
pres2=0
adcon1=6 (digital)
estado=0
estado2=0
pres1=0
No
estado.0=0
Si
estado.0=1
pres2=0
No
estado.1=0
Si
estado.1=1
senx1=0
No
estado.3=0
Si
estado.3=1
senx2=0
No
estado.4=0
Si
estado.4=1
humo=0
No
estado.5=1
Si
estado.5=0
Envió información
estado
XX estado ZZ
1


1
h1=0
No
estado2.0=0
Si
estado2.0=1
h2=0
No
estado2.1=0
Si
estado2.1=1
h3=0
No
Estado2.2=0
Si
estado2.2=1
h4=0
No
estado2.3=0
Si
estado2.=1
h5=0
No
estado2.4=0
Si
estado2.4=1
h6=0
No
estado2.5=0
Si
estado2.5=1
h7=0
No
estado2.6=0
Si
estado2.6=1
h8=0
No
estado2.7=0
Si
estado2.7=1
2

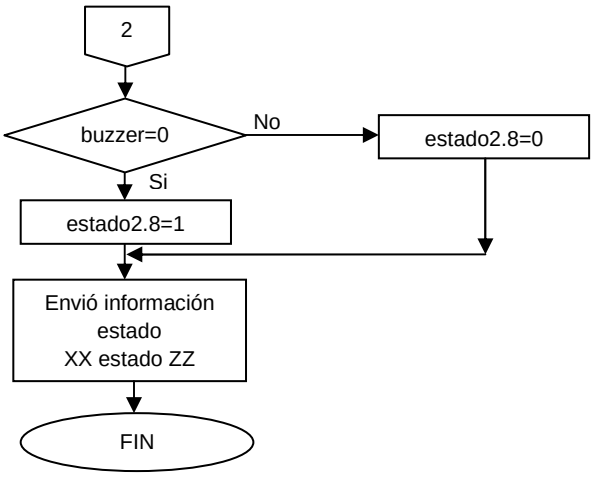
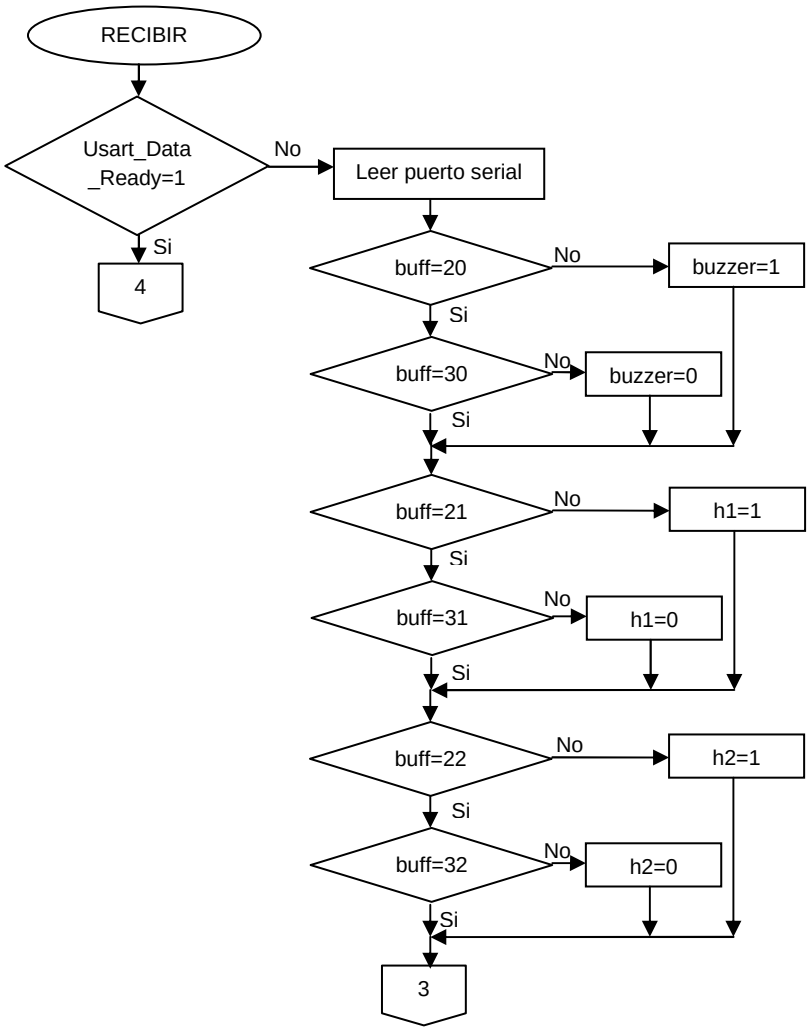


Figura 3.14: Diagrama de flujo de la subrutina estados.



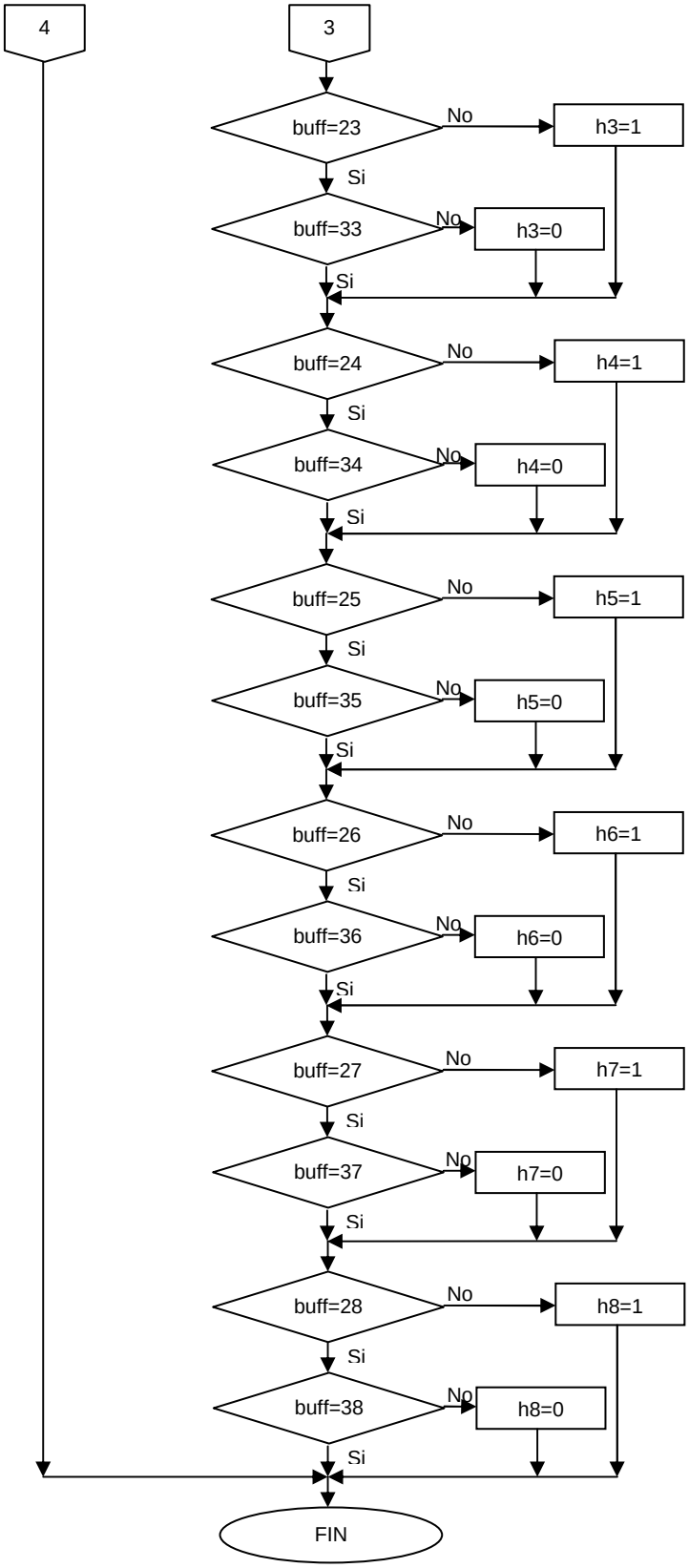
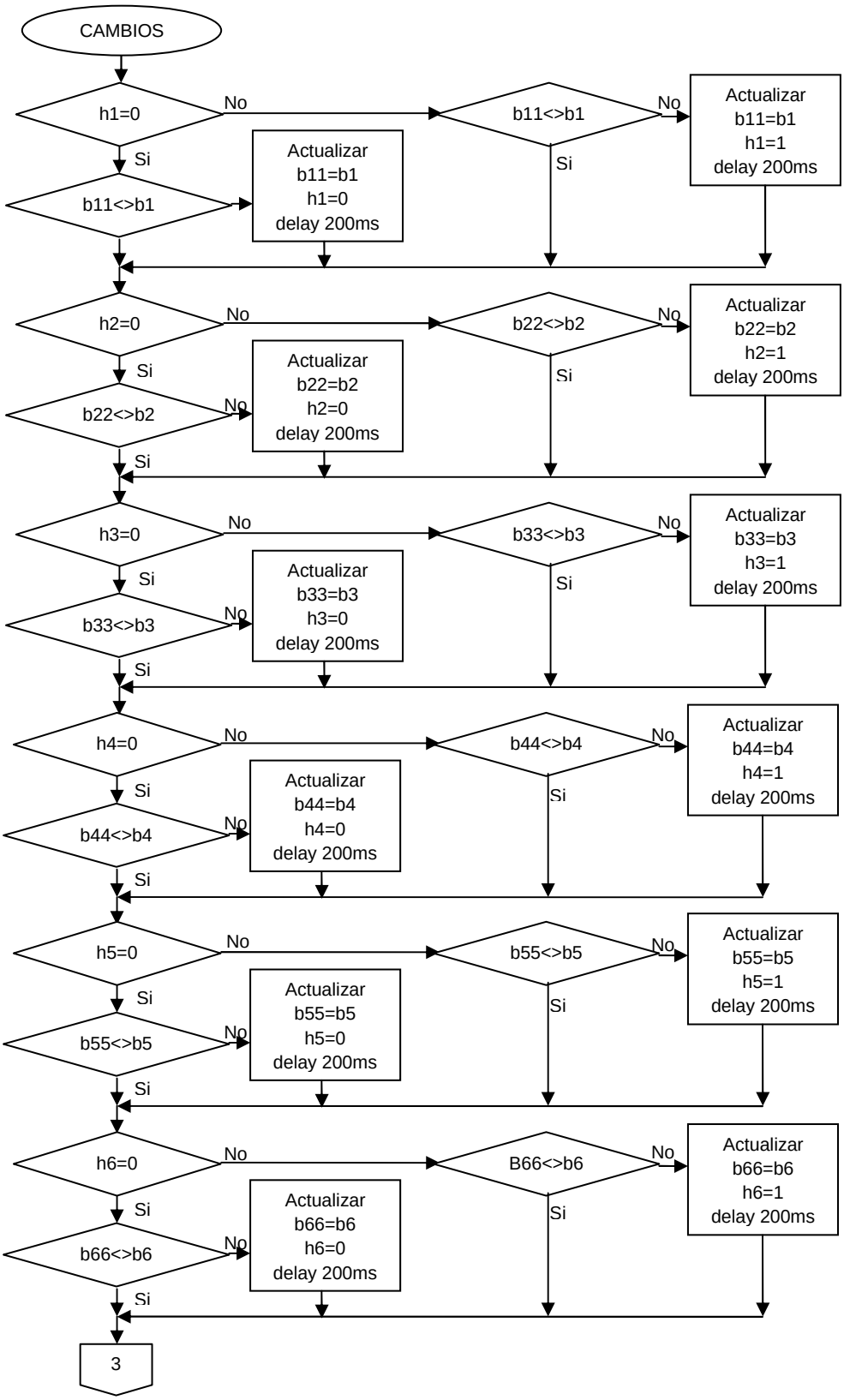


Figura 3.15: Diagrama de flujo de la subrutina recibir.


CAMBIOS
h1=0
No
b11<>b1
No
Actualizar
b11=b1
h1=1
delay 200ms
Si
Si
b11<>b1
Actualizar
b11=b1
h1=0
delay 200ms
h2=0
No
b22<>b2
No
Actualizar
b22=b2
h2=1
delay 200ms
Si
Si
b22<>b2
No
Actualizar
b22=b2
h2=0
delay 200ms
Si
h3=0
No
b33<>b3
No
Actualizar
b33=b3
h3=1
delay 200ms
Si
Si
b33<>b3
No
Actualizar
b33=b3
h3=0
delay 200ms
Si
h4=0
No
b44<>b4
No
Actualizar
b44=b4
h4=1
delay 200ms
Si
Si
b44<>b4
No
Actualizar
b44=b4
h4=0
delay 200ms
Si
h5=0
No
b55<>b5
No
Actualizar
b55=b5
h5=1
delay 200ms
Si
Si
b55<>b5
No
Actualizar
b55=b5
h5=0
delay 200ms
Si
h6=0
No
B66<>b6
No
Actualizar
b66=b6
h6=1
delay 200ms
Si
Si
b66<>b6
No
Actualizar
b66=b6
h6=0
delay 200ms
Si
3

`

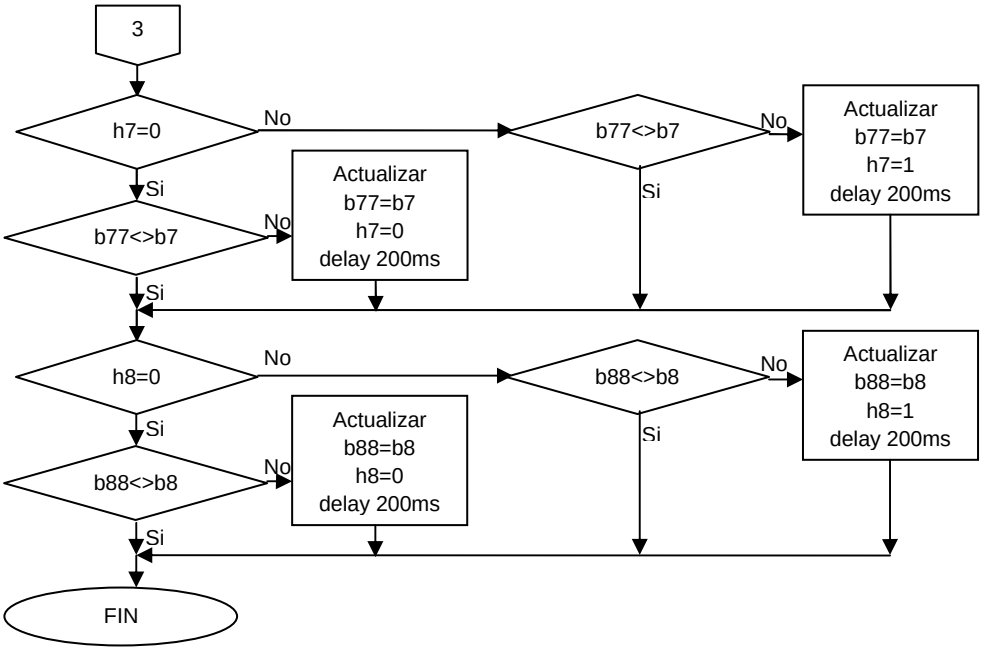


Figura 3.16: Diagrama de flujo de la subrutina cambios

CÓDIGO FUENTE DE MIKROBASIC

program CasaAlarma

symbol pres1=portb.1 symbol senx1=portb.2 symbol senx2=portb.3 symbol senx3=portb.4 symbol humo=portb.5

symbol humo=portb.5 symbol h1=portd.7 symbol h2=portd.6 symbol h3=portd.5 symbol h4=portd.4

symbol h5=portd.3 symbol h6=portd.2 symbol h7=portd.1 symbol h8=portd.0 symbol buzzer=portc.3

symbol b1=porta.0 symbol b2=porta.1 symbol b3=porta.2 symbol b4=porta.3 symbol b5=porta.4 symbol b6=porta.5 symbol b7=porte.0 symbol b8=porte.1

dim buff as byte dim estado as byte dim estado2 as word

dim txt as string[10]

dim b11 as boolean dim b22 as boolean dim b33 as boolean dim b44 as boolean dim b55 as boolean dim b66 as boolean dim b77 as boolean dim b88 as boolean

```
dim pres2 as byte
dim dato as word
dim datof as float
dim i as byte
sub procedure estados()
   adcon1=0
   dato=adc_read(7)
   datof=dato*5/1023
   if datof<2.5 then
      pres2=1
   else
      pres2=0
   end if
   estado=0
   estado2=0
   adcon1=6
   if pres1=0 then estado.0=0 end if
   if pres2=0 then estado.1=0 end if
   if senx1=0 then estado.2=0 end if
   if senx2=0 then estado.3=0 end if
   if senx3=0 then estado.4=0 end if
   if humo=0 then estado.5=1 end if
   if pres1=1 then estado.0=1 end if
   if pres2=1 then estado.1=1 end if
   if senx1=1 then estado.2=1 end if
   if senx2=1 then estado.3=1 end if
   if senx3=1 then estado.4=1 end if
   if humo=1 then estado.5=0 end if
   bytetostr(estado,txt)
   usart_write_text("XX")
   usart_write_text(txt)
   usart_write_text("ZZ")
   usart_write(13)
```

```
    if h1 =0 then estado2.0=0 end if
    if h2 =0 then estado2.1=0 end if
    if h3 =0 then estado2.2=0 end if
    if h4 =0 then estado2.3=0 end if
    if h5 =0 then estado2.4=0 end if
    if h6 =0 then estado2.5=0 end if
    if h7 =0 then estado2.6=0 end if
    if h8 =0 then estado2.7=0 end if
    if buzzer=0 then estado2.8=0 end if
    if h1 =1 then estado2.0=1 end if
    if h2 =1 then estado2.1=1 end if
    if h3 =1 then estado2.2=1 end if
    if h4 =1 then estado2.3=1 end if
    if h5 =1 then estado2.4=1 end if
    if h6 =1 then estado2.5=1 end if
    if h7 =1 then estado2.6=1 end if
    if h8 =1 then estado2.7=1 end if
    if buzzer=1 then estado2.8=1 end if
    wordtostr(estado2,txt)
    usart_write_text("YY")
    usart_write_text(txt)
    usart_write_text("ZZ")
    usart_write(13)
end sub
sub procedure recibir()
    if Usart_Data_Ready = 1 then
      buff = Usart_Read
      usart_write(Usart_Read)
      if buff=20 then buzzer=1 end if
      if buff=21 then h1=1 end if
      if buff=22 then h2=1 end if
      if buff=23 then h3=1 end if
      if buff=24 then h4=1 end if
```

```
      if buff=25 then h5=1 end if
      if buff=26 then h6=1 end if
      if buff=27 then h7=1 end if
      if buff=28 then h8=1 end if
      if buff=30 then buzzer=0 end if
      if buff=31 then h1=0 end if
      if buff=32 then h2=0 end if
      if buff=33 then h3=0 end if
      if buff=34 then h4=0 end if
      if buff=35 then h5=0 end if
      if buff=36 then h6=0 end if
      if buff=37 then h7=0 end if
      if buff=38 then h8=0 end if
    end if
end sub
main:
   trisa=255
   trise.0=1 trise.1=1
   adcon1=6
   trisb=255 option_reg=0
   trisd=0 portd=255
   trisc=0
   Usart_Init(1200)
   buzzer=1
   usart_write_text("inicio")
   portd=255
   delay_ms(1000)
   buzzer=0  b11=0
   b11=b1
   b11=b1 b22=b2 b33=b3 b44=b4
   b55=b5 b66=b6 b77=b7 b88=b8
   while true
      estados
```

```
recibir
if h1=0 then
  if b11 <> b1 then
    b11=b1
    h1=1 delay_ms(200)
  end if
else
  if b11 <> b1 then
    b11=b1
    h1=0 delay_ms(200)
  end if
end if

if h2=0 then
  if b22 <> b2 then
    b22=b2
    h2=1 delay_ms(200)
  end if
else
  if b22 <> b2 then
    b22=b2
    h2=0 delay_ms(200)
  end if
end if

if h3=0 then
  if b33 <> b3 then
    b33=b3
    h3=1 delay_ms(200)
  end if
else
  if b33 <> b3 then
    b33=b3
```

```
      h3=0 delay_ms(200)
    end if
end if

if h4=0 then
    if b44 <> b4 then
      b44=b4
      h4=1 delay_ms(200)
    end if
else
    if b44 <> b4 then
      b44=b4
      h4=0 delay_ms(200)
    end if
end if

if h5=0 then
    if b55 <> b5 then
      b55=b5
      h5=1 delay_ms(200)
    end if
else
    if b55 <> b5 then
      b55=b5
      h5=0 delay_ms(200)
    end if
end if

if h6=0 then
    if b66 <> b6 then
      b66=b6
      h6=1 delay_ms(200)
    end if
```

```
    else
      if b66 <> b6 then
        b66=b6
        h6=0 delay_ms(200)
      end if
    end if

    if h7=0 then
      if b77 <> b7 then
        b77=b7
        h7=1 delay_ms(200)
      end if
    else
      if b77 <> b7 then
        b77=b7
        h7=0 delay_ms(200)
      end if
    end if

    if h8=0 then
      if b88 <> b8 then
        b88=b8
        h8=1 delay_ms(200)
      end if
    else
      if b88 <> b8 then
        b88=b8
        h8=0 delay_ms(200)
      end if
    end if
  wend
end.
```

3.1.6.2. Pantalla programa principal

Esta pantalla es la principal en la que va a interectuar el usuario, tiene por objetivo controlar y visualizar las luminarias, los sensores y activar el sistema de alarma.

Esta pantalla corresponde a la forma real del prototipo realizado en este proyecto.

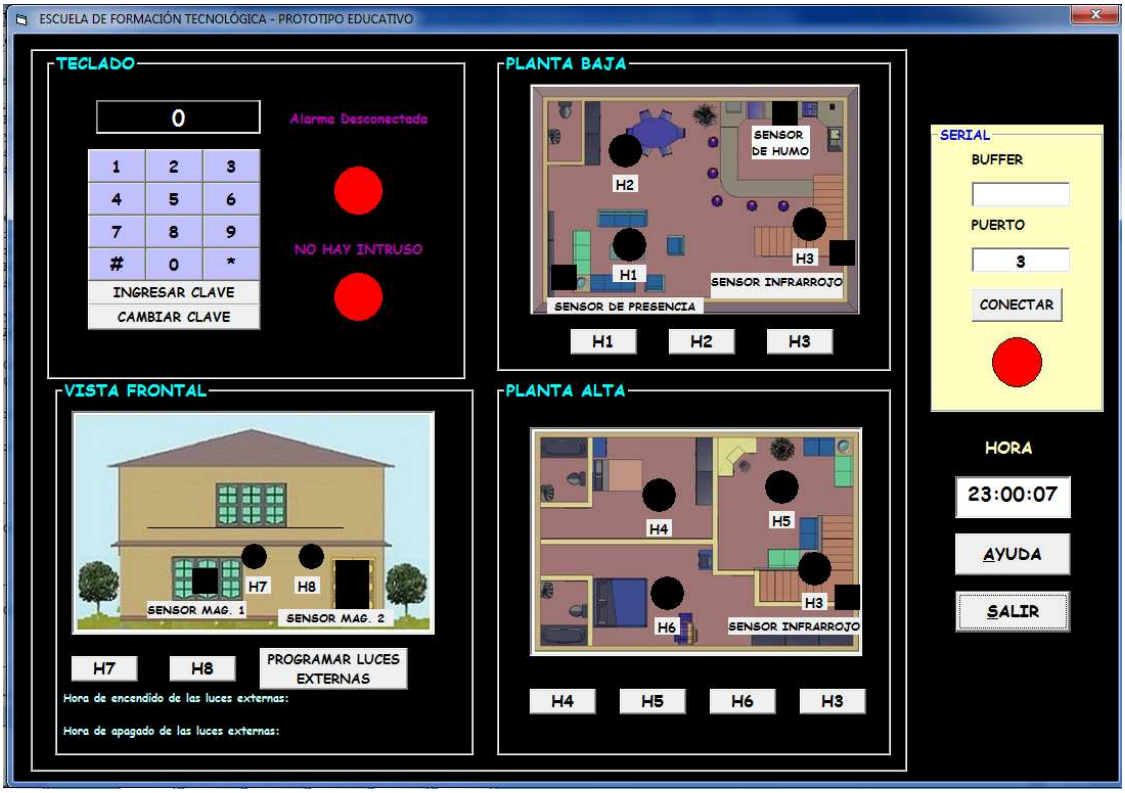


Figura 3.17: Interfaz del programa de control de Visual Basic.

CÓDIGO FUENTE DE VISUAL BASIC

FORMULARIO 1

```
Dim DATO As Byte
Dim NUM As Byte
Dim Buffer As String
Dim Puerto As Byte
Dim Conectar As Boolean
Dim a, b, c, e, f As Byte
```

```
Dim d As Integer
Dim g As String
Dim pres1, pres2, senx1, senx2, senx3, humo As Byte
Dim h1, h2, h3, h4, h5, h6, h7, h8, buzzer As Byte
Dim primer As Byte
Dim contador As Byte
Dim var As Byte
Dim PRIMERA_ALARMA As Boolean
Private Sub Command1_Click()
If Alarma = 0 Then
 If h4 Then
   enviar Chr(34)
 Else
   enviar Chr(24)
 End If
End If
End Sub
Private Sub Command10_Click()
 DATO = 2
 Claves
End Sub
Private Sub Command11_Click()
 DATO = 3
 Claves
End Sub
Private Sub Command12_Click()
 DATO = 4
 Claves
End Sub
Private Sub Command13_Click()
 DATO = 5
 Claves
End Sub
```

```
Private Sub Command14_Click()
 DATO = 6
 Claves
End Sub
Private Sub Command15_Click()
 DATO = 7
 Claves
End Sub
Private Sub Command16_Click()
 DATO = 8
 Claves
End Sub
Private Sub Command17_Click()
 DATO = 9
 Claves
End Sub
Private Sub Command18_Click()
Clave = 0
NUM = 0
Text1.Text = Clave
End Sub
Private Sub Command19_Click()
 DATO = 0
 Claves
End Sub
Private Sub Command2_Click()
If Alarma = 0 Then
 If h5 Then
   enviar Chr(35)
 Else
   enviar Chr(25)
 End If
End If
```

```
End Sub
Private Sub Command20_Click()
 Clave = 0
 NUM = 0
 Text1.Text = Clave
End Sub
Private Sub Command21_Click()
 If Clave = ClaveFinal Then
    cnt = 5
    Form3.Label1.Caption = 5
    Form3.Timer1.Enabled = True
    Form3.Show
 Else
 End If
 Clave = 0
 NUM = 0
 Text1.Text = ""
End Sub
Private Sub Command22_Click()
 End
End Sub
Private Sub Command23_Click()
If Alarma = 0 Then
 If h8 Then
   enviar Chr(38)
 Else
   enviar Chr(28)
 End If
End If
End Sub
Private Sub Command24_Click()
 Form2.Show
End Sub
```

```
Private Sub Command25_Click()
enviar Chr(30)
End Sub
Private Sub Command26_Click()
End Sub
Private Sub Command3_Click()
If Alarma = 0 Then
 If h6 Then
   enviar Chr(36)
 Else
   enviar Chr(26)
 End If
End If
End Sub
Private Sub Command4_Click()
If Alarma = 0 Then
 If h3 Then
   enviar Chr(33)
 Else
   enviar Chr(23)
 End If
End If
End Sub
Private Sub Command5_Click()
If Alarma = 0 Then
 If h7 Then
   enviar Chr(37)
 Else
   enviar Chr(27)
 End If
End If
End Sub
Private Sub Command6_Click()
```

```
If Alarma = 0 Then
 If h3 Then
   enviar Chr(33)
 Else
   enviar Chr(23)
 End If
End If
End Sub
Private Sub Command7_Click()
If Alarma = 0 Then
 If h2 Then
   enviar Chr(32)
 Else
   enviar Chr(22)
 End If
End If
End Sub
Private Sub Command8_Click()
If Alarma = 0 Then
 If h1 Then
   enviar Chr(31)
 Else
   enviar Chr(21)
 End If
End If
End Sub
Private Sub Command9_Click()
 DATO = 1
 Claves
End Sub
Private Sub Claves()
   NUM = NUM + 1
   If NUM = 1 Then Clave = DATO
```

```
   If NUM = 2 Or NUM = 3 Or NUM = 4 Then
      Clave = Clave * 10
      Clave = Clave + DATO
      If NUM = 4 Then NUM = 0
   End If
   Text1.Text = Clave
End Sub
Private Sub CONN_Click()
  If Conectar Then
    cerrar
    Conectar = False
  Else
    abrir
    Conectar = True
  End If
End Sub
Private Sub Form_Load()
  PRIMERA_ALARMA = False
  Label3.Caption = "Vacío"
  BUZZ.FillColor = vbBlack
  ClaveFinal = 1234
  Puerto = 3
  CONN.Caption = "CONECTAR"
  Text3.Text = Puerto
  Timer3.Enabled = False
End Sub
Public Function leer() As String
    Dim st As String
    If MSComm1.InBufferCount <> 0 Then
        Buffer = MSComm1.Input
    End If
End Function
Public Sub abrir()
```

```
    If Not MSComm1.PortOpen Then
        Puerto = Val(Text3.Text)
        MSComm1.CommPort = Puerto
        MSComm1.Settings = "1200" & ",n,8,1"
        MSComm1.InputLen = 0
        MSComm1.PortOpen = True
    Else
        MsgBox ("Puerto COM 1 está abierto"), , "Puerto de Comunicaciones Serial"
    End If
Exit Sub
mierror:
    MsgBox "Número de Puerto NO VALIDO", vbInformation
    numeroPuerto = CInt(InputBox("Escriba el número de Puerto de
Comunicación", "Puerto de Comunicación", "1"))
    abrir
End Sub
Public Function enviar(cadenaSalida As String)
On Error GoTo mierror
Dim i As String
    If MSComm1.PortOpen = False Then
        MsgBox "El Puerto está cerrado, ábralo "
    Else
        MSComm1.Output = cadenaSalida & Chr(&HD)
    End If
Exit Function
mierror:
    MsgBox "Error Inesperado con el Puerto"
End Function
Public Sub cerrar()
If MSComm1.PortOpen Then
    MSComm1.PortOpen = False
 Else
```

```
   MsgBox ("Puerto COM" & Puerto & " está cerrado"), , "Puerto de
Comunicaciones Serial"
End If
End Sub
Private Sub PROG_Click()
 Form4.Show
End Sub
Private Sub Timer1_Timer()
Dim HORA1 As Byte
Dim HORA2 As Byte
Dim MINUTO1 As Byte
Dim MINUTO2 As Byte
Dim HORA_AUX As Byte
Dim MINUTO_AUX As Byte
HORA1 = Val(Mid(HORA_INICIO, 1, 2))
MINUTO1 = Val(Mid(HORA_INICIO, 4, 2))
HORA2 = Val(Mid(HORA_FIN, 1, 2))
MINUTO2 = Val(Mid(HORA_FIN, 4, 2))
HORA_AUX = Val(Mid(Time$, 1, 2))
MINUTO_AUX = Val(Mid(Time$, 4, 2))
Text7.Text = MINUTO_AUX
 Text6.Text = Time$
 Label22.Caption = "H.inicio= " & HORA_INICIO
 Label23.Caption = "H.fin   = " & HORA_FIN
 Label24.Caption = "F.inicio= " & FECHA_INICIO
 Label25.Caption = "F.fin   = " & FECHA_FIN
 If Alarma = 1 Then
  Label4.Caption = "Alarma Conectada"
  Shape2.FillColor = vbGreen
  If Conectar Then
     If pres1 Or pres2 Or senx1 Or senx2 Or senx3 Or humo Then
       enviar Chr(20)
       Label3.Caption = "Intruso"
```

```
    End If
  End If
  PROG.Enabled = False
  If FECHA_INICIO = Date$ And HORA_INICIO = Time$ Then
    enviar Chr(38)
    PRIMERA_ALARMA = True
    Timer4.Enabled = True
  End If
  If FECHA_FIN = Date$ And HORA_FIN = Time$ Then
    enviar Chr(28)
    PRIMERA_ALARMA = False
    Timer4.Enabled = True
  End If
  Else
  PROG.Enabled = True
  Label4.Caption = "Alarma Desconectada"
  Shape2.FillColor = vbRed
 End If
 If Conectar Then
  Shape1.FillColor = vbGreen
  leer
  If Buffer <> "" Then
    Text2.Text = Buffer
    For a = 1 To 10
      If Mid(Buffer, a, 2) = "YY" Then
        For b = a To 20
          If Mid(Buffer, b, 2) = "ZZ" Then
            If b > a Then
              c = b - a - 2
              g = Mid(Buffer, a + 2, c)
              d = Val(g)
              compara1
            End If
```

```
            Exit For
          End If
        Next b
        Exit For
      End If

      If Mid(Buffer, a, 2) = "XX" Then
        For b = a To 10
          If Mid(Buffer, b, 2) = "ZZ" Then
            If b > a Then
              c = b - a - 2
              g = Mid(Buffer, a + 2, c)
              d = Val(g)
              compara2
            End If
            Exit For
          End If
        Next b
        Exit For
      End If
    Next a
  End If
 Else
  Shape1.FillColor = vbRed
End If
End Sub
Private Sub compara1()
              If d And 1 Then
                h1 = 1
                H11.FillColor = vbBlack
              Else
                h1 = 0
                H11.FillColor = vbYellow
```

```
End If
If d And 2 Then
  h2 = 1
  H22.FillColor = vbBlack
Else
  h2 = 0
  H22.FillColor = vbYellow
End If
If d And 4 Then
  h3 = 1
  H33.FillColor = vbBlack
  H77.FillColor = vbBlack
Else
  h3 = 0
  H33.FillColor = vbYellow
  H77.FillColor = vbYellow
End If
If d And 8 Then
  h4 = 1
  H44.FillColor = vbBlack
Else
  h4 = 0
  H44.FillColor = vbYellow
End If
If d And 16 Then
  h5 = 1
  H55.FillColor = vbBlack
Else
  h5 = 0
  H55.FillColor = vbYellow
End If
If d And 32 Then
  h6 = 1
```

```
            H66.FillColor = vbBlack
          Else
            h6 = 0
            H66.FillColor = vbYellow
          End If
          If d And 64 Then
            h7 = 1
            H88.FillColor = vbBlack
          Else
            h7 = 0
            H88.FillColor = vbYellow
          End If
          If d And 128 Then
            h8 = 1
            H99.FillColor = vbBlack
          Else
            h8 = 0
            H99.FillColor = vbYellow
          End If
          If d And 256 Then
            buzzer = 1
            BUZZ.FillColor = vbBlack
          Else
            buzzer = 0
            BUZZ.FillColor = vbYellow
          End If
End Sub
Private Sub compara2()
          If d And 1 Then
            pres1 = 1
            pres11.FillColor = vbRed
          Else
            pres1 = 0
```

```
  pres11.FillColor = vbBlack
End If
If d And 2 Then
  pres2 = 0
  pres22.FillColor = vbBlack
  Shape3.FillColor = vbBlack
  If primer = 0 Then
     primer = 1
     enviar Chr(23)
  Timer2.Enabled = False
  End If
Else
  pres2 = 1
  pres22.FillColor = vbRed
  Shape3.FillColor = vbRed
  If primer = 1 Then
    primer = 2
    Timer2.Enabled = True
    enviar Chr(33)
  End If
End If
If d And 4 Then
  senx1 = 0
  senx11.FillColor = vbRed
Else
  senx1 = 0
  senx11.FillColor = vbBlack
End If
If d And 8 Then
  senx2 = 1
  senx22.FillColor = vbRed
Else
  senx2 = 0
```

```
            senx22.FillColor = vbBlack
           End If
           If d And 16 Then
            senx3 = 1
            senx33.FillColor = vbRed
           Else
            senx3 = 0
            senx33.FillColor = vbBlack
           End If
           If d And 32 Then
            humo = 1
            humo1.FillColor = vbRed
           Else
            humo = 0
            humo1.FillColor = vbBlack
           End If
          Text4.Text = humo & " " & senx3 & " " & senx2 & " " & senx1 & " " &
pres2 & " " & pres1
End Sub
Private Sub Timer2_Timer()
enviar Chr(23)
Timer2.Enabled = False
primer = 0
End Sub
Private Sub Timer3_Timer()
 Dim NUM As Double
 contador = contador + 1
 If contador = 1 Then
  Randomize
  NUM = Rnd * 255
  var = NUM
 End If
 Text5.Text = var & "  " & contador
```

```
If contador = 1 Then
 If var And 1 Then
  enviar Chr(21)
 Else
  enviar Chr(31)
 End If
End If
If contador = 2 Then
 If var And 2 Then
  enviar Chr(22)
 Else
  enviar Chr(32)
 End If
End If
If contador = 3 Then
 If var And 4 Then
  enviar Chr(23)
 Else
  enviar Chr(33)
 End If
End If
If contador = 4 Then
 If var And 8 Then
  enviar Chr(24)
 Else
  enviar Chr(34)
 End If
End If
If contador = 5 Then
 If var And 16 Then
  enviar Chr(25)
 Else
  enviar Chr(35)
```

```
 End If
End If
If contador = 6 Then
 If var And 32 Then
  enviar Chr(26)
 Else
  enviar Chr(36)
 End If
End If
If contador = 7 Then
 If var And 64 Then
  enviar Chr(27)
 Else
  enviar Chr(37)
 End If
End If
If contador = 8 Then
 If var And 128 Then
  enviar Chr(28)
 Else
  enviar Chr(38)
 End If
End If
If contador = 50 Then contador = 0
End Sub
Private Sub Timer4_Timer()
 Dim i As Byte
  If PRIMERA_ALARMA = False Then
   For i = 1 To 100
    enviar Chr(27)
   Next i
  Else
   For i = 1 To 100
```

```
      enviar Chr(37)
    Next i
   End If
   Timer4.Enabled = False
End Sub
```

FORMULARIO 2

```
Dim NUM As Byte
Dim DATO As Byte
Dim PRIMERA As Byte
Private Sub CmdLimpiar_Click()
Text1.Text = ""
Text1.SetFocus
End Sub
Private Sub CmdSalir_Click()
Unload Me
End Sub
Private Sub Command19_Click()
 DATO = 0
 Claves
End Sub
Private Sub Command21_Click()
   If PRIMERA = 1 Then
     MsgBox "SU NUEVA CLAVE ES " & Clave, vbInformation
     ClaveFinal = Clave
     PRIMERA = 0
     Unload Me
     Exit Sub
  End If
 If Clave = ClaveFinal Then
  If PRIMERA = 0 Then
     MsgBox "INGRESE SU NUEVA CLAVE", vbInformation
```

```
    PRIMERA = 1
  End If
 Else
  PRIMERA = 0
  Unload Me
  Exit Sub
 End If
 Clave = 0
 NUM = 0
 Text1.Text = ""
End Sub
Private Sub Command9_Click()
 DATO = 1
 Claves
End Sub
Private Sub Command10_Click()
 DATO = 2
 Claves
End Sub
Private Sub Command11_Click()
 DATO = 3
 Claves
End Sub
Private Sub Command12_Click()
 DATO = 4
 Claves
End Sub
Private Sub Command13_Click()
 DATO = 5
 Claves
End Sub
Private Sub Command14_Click()
 DATO = 6
```

```
 Claves
End Sub
Private Sub Command15_Click()
 DATO = 7
 Claves
End Sub
Private Sub Command16_Click()
 DATO = 8
 Claves
End Sub
Private Sub Command17_Click()
 DATO = 9
 Claves
End Sub
Private Sub Command18_Click()
Clave = 0
NUM = 0
Text1.Text = Clave
End Sub
Private Sub Claves()
   NUM = NUM + 1
   If NUM = 1 Then Clave = DATO
   If NUM = 2 Or NUM = 3 Or NUM = 4 Then
      Clave = Clave * 10
      Clave = Clave + DATO
      If NUM = 4 Then NUM = 0
   End If
   Text1.Text = Clave
End Sub
Private Sub Form_Load()
MsgBox "INGRESE LA CLAVE ACTUAL", vbInformation
PRIMERA = 0
End Sub
```

FORMULARIO 3

```
Private Sub Form_Load()
 cnt = 5
 Label1.Caption = cnt
End Sub
Private Sub Timer1_Timer()
If cnt > 0 Then
   cnt = cnt - 1
Else
   If Alarma Then
      Alarma = 0
      Form1.enviar Chr(30)
      Form1.Label3.Caption = "Vacío"
      Form1.Command24.Enabled = True
      Form1.Timer3.Enabled = False
      Form1.enviar Chr(30)
      Timer2.Enabled = True
    Else
      Form1.Command24.Enabled = False
      Alarma = 1
       'Form1.Timer3.Enabled = True
      Unload Me
   End If
   cnt = 5
   Label1.Caption = cnt
   Timer1.Enabled = False
End If
Label1.Caption = cnt
End Sub
Private Sub Timer2_Timer()
    Form1.enviar Chr(30)
    Timer2.Enabled = False
```

```
    Unload Me
End Sub
```

MÓDULO 1

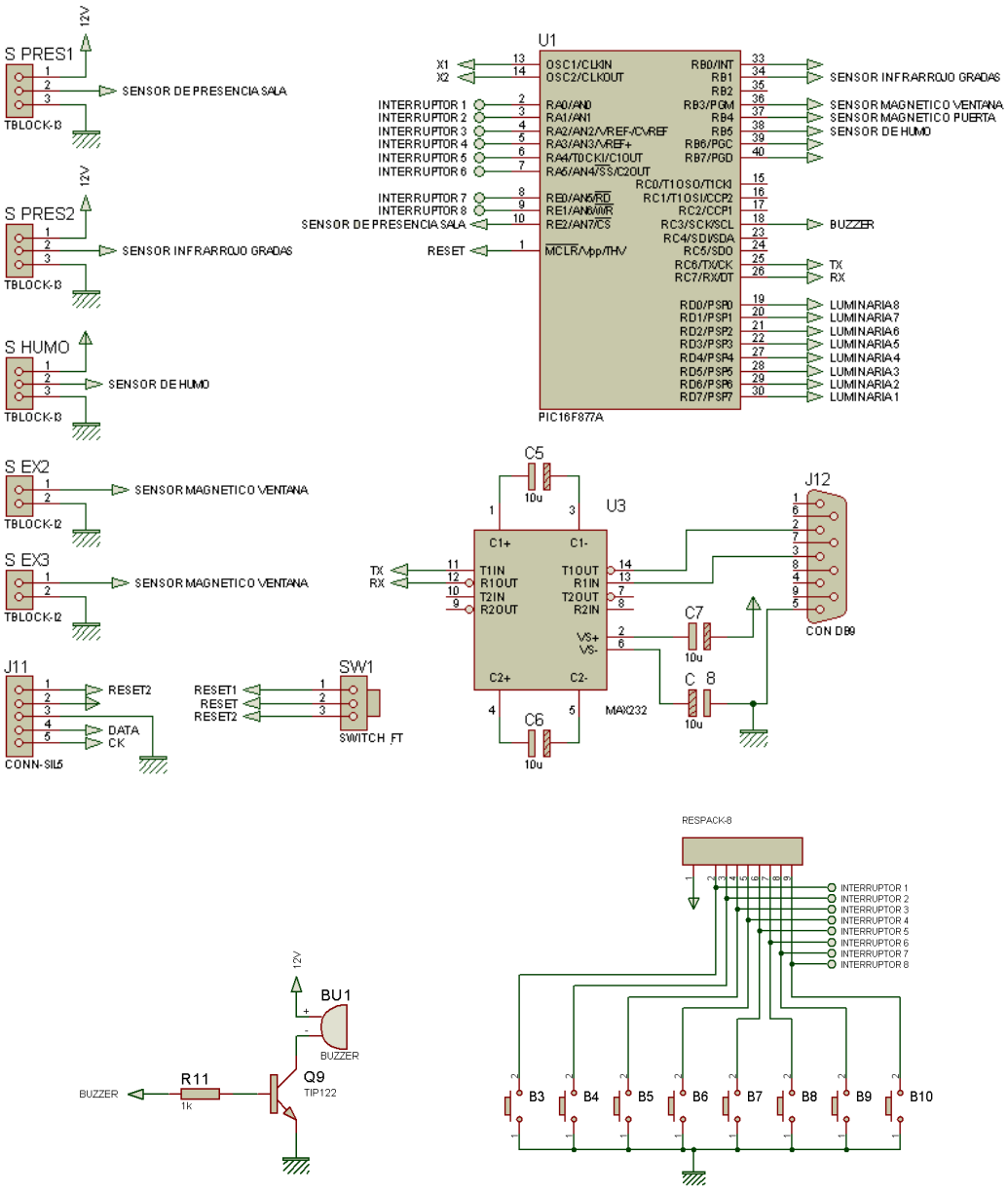
```
Global Clave As Integer
Global ClaveFinal As Integer
Global Alarma As Byte
Global cnt As Byte
Global HORA_INICIO As String
Global HORA_FIN As String
Global FECHA_INICIO As String
Global FECHA_FIN As String
```

CAPÍTULO IV:
PRUEBAS Y PLACAS

4.1. DIAGRAMA CIRCUITAL Y DE PISTAS

4.1.1. DISEÑO DE LA TARJETA ELECTRÓNICA.

4.1.1.1. Diagrama circuital



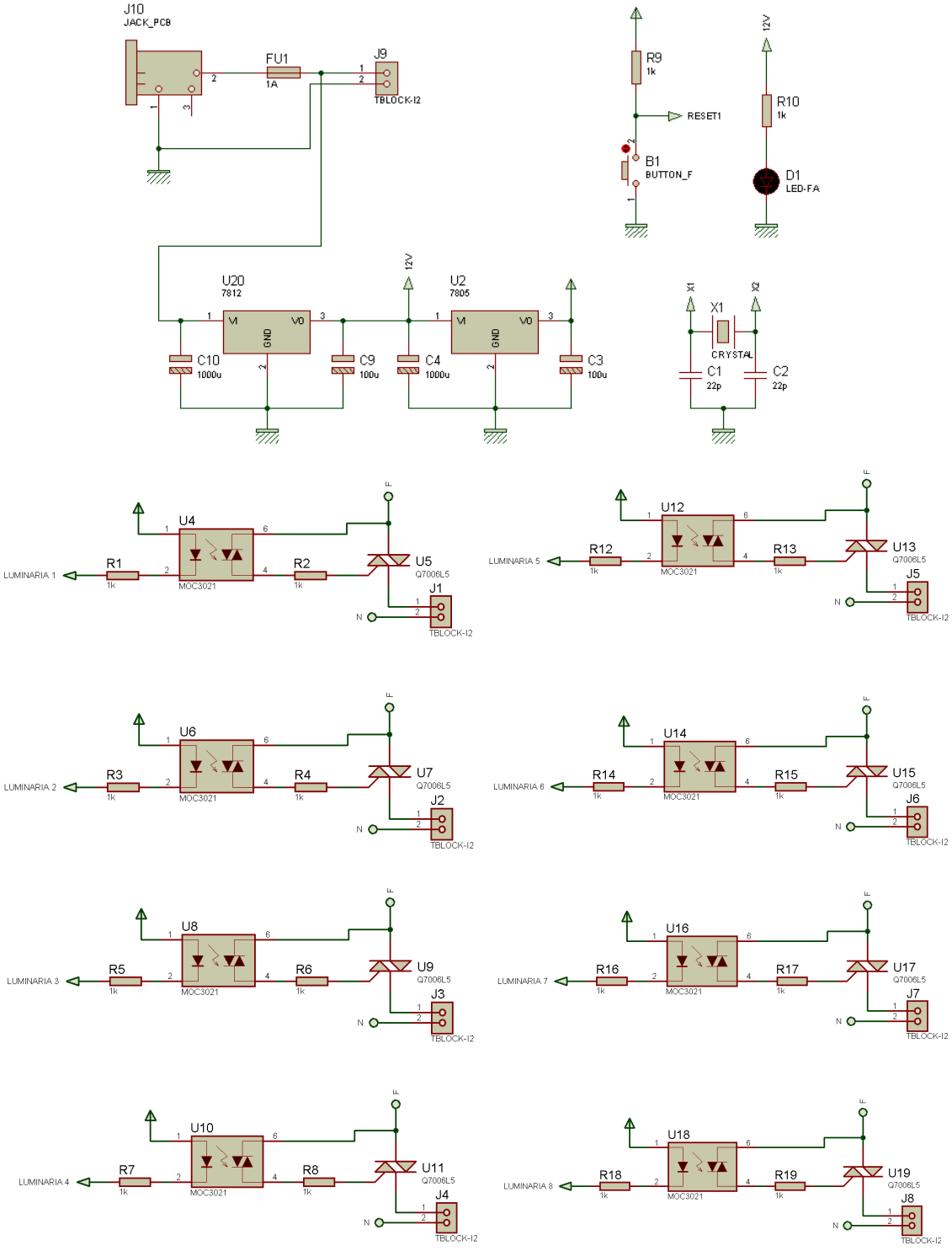


Figura 4.1: Diseño del diagrama circuital.

4.1.1.2. Diagrama de pistas

Para realizar nuestro proyecto necesitamos la placa de cobre en donde imprimiremos las pistas de nuestro circuito como indica la Figura 3.13:

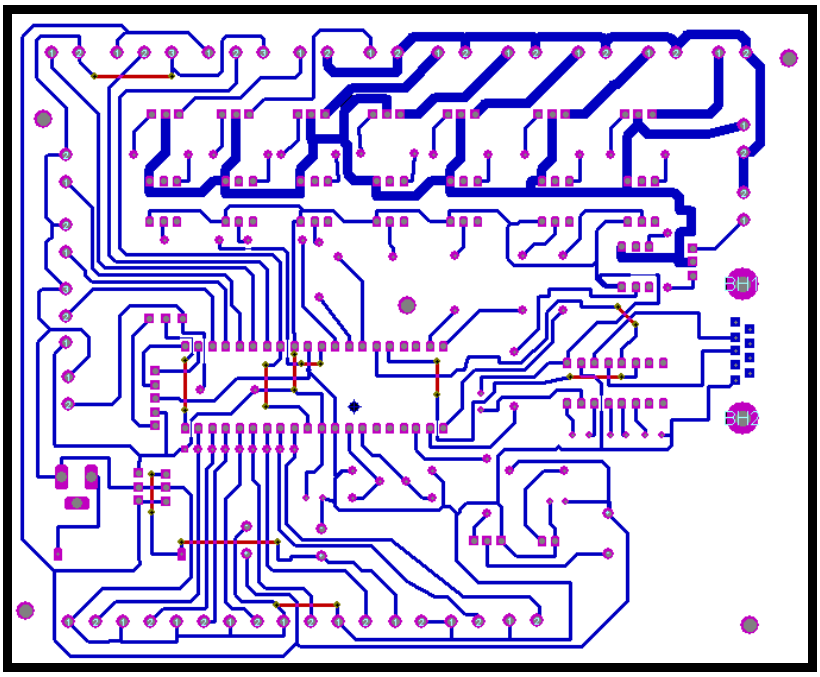


Figura 4.2: Diagrama de pistas.

4.1.1.3. Diagrama posicional de elementos

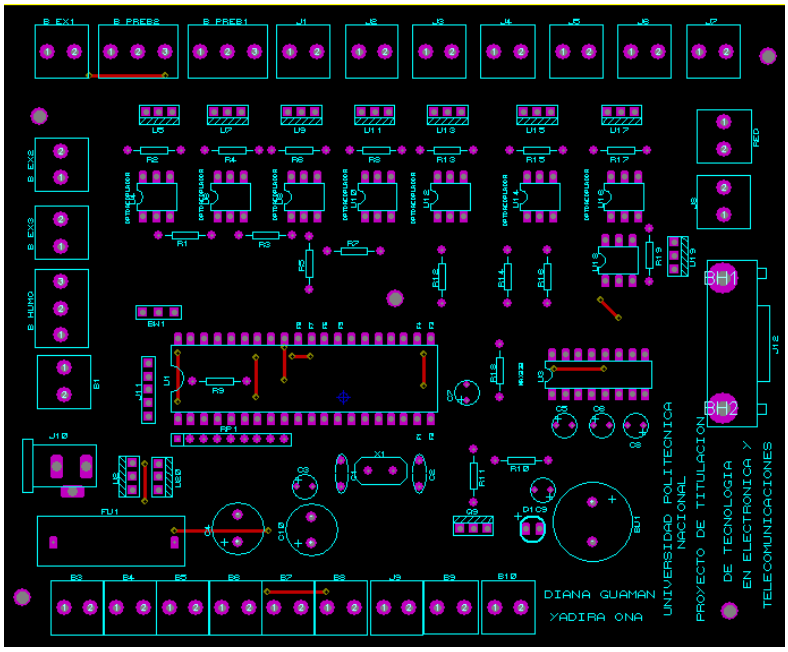


Figura 4.3: Diagrama posicional de elementos.

Visualización 3D

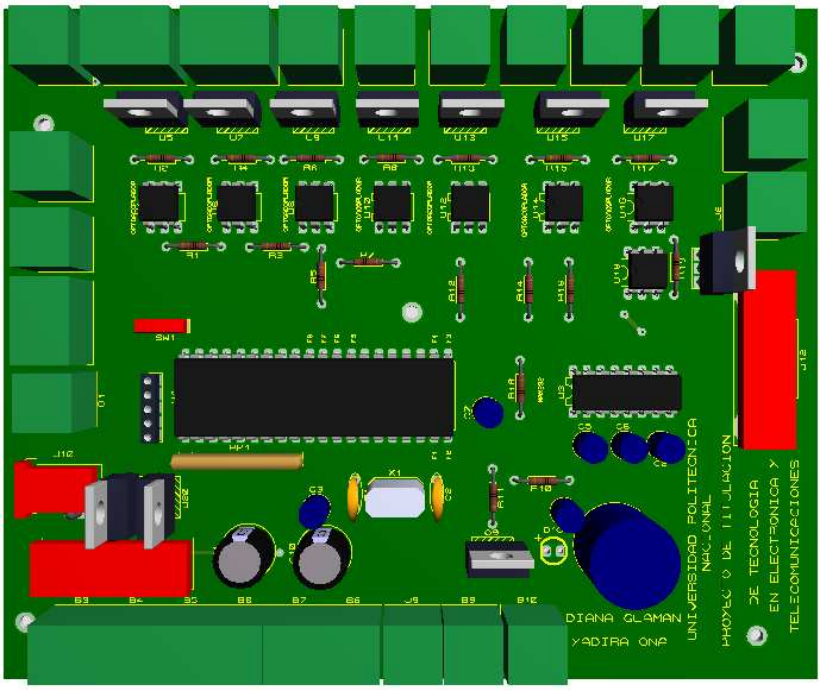


Figura 4.4: Diagrama de la posición de los elementos en 3D.

4.2. MONTAJE EN EL PROTOTIPO

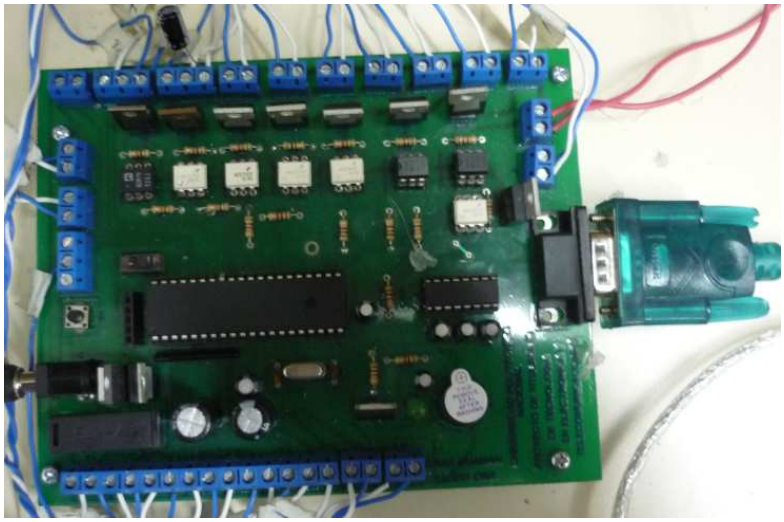


Figura 4.5: Vista superior de la tarjeta.

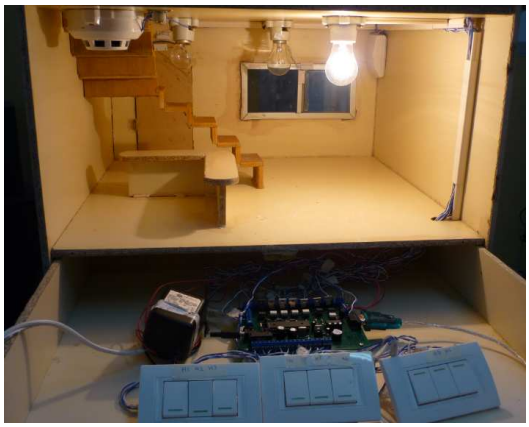
PROTOTIPO	PROGRAMA EN VISUAL BASIC 6.0
	

Tabla 4.1: Monitoreo de la luminaria H2.

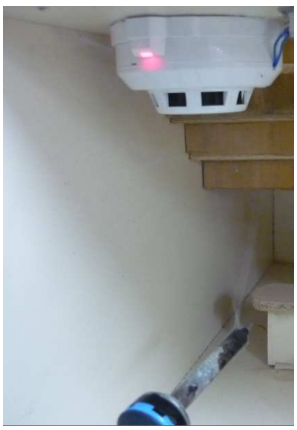
PROTOTIPO	PROGRAMA EN VISUAL BASIC 6.0
	

Tabla 4.2: Monitoreo del sensor de humo.

PROTOTIPO	PROGRAMA EN VISUAL BASIC 6.0
	

Tabla 4.3: Monitoreo del sensor de presencia.

PROTOTIPO	PROGRAMA EN VISUAL BASIC 6.0
	

Tabla 4.4: Monitoreo del sensor infrarrojo cuando detecta presencia.

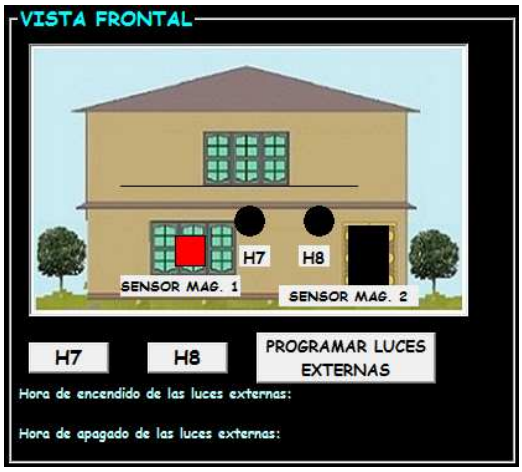
PROTOTIPO	PROGRAMA EN VISUAL BASIC 6.0
Vista frontal de la ventana en donde se encuentra el sensor magnético 1  Vista interior de la ventana en donde se encuentra el sensor magnético 1 	

Tabla 4.5: Monitoreo del sensor magnético 1.

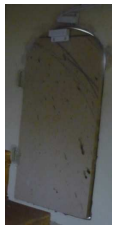
PROTOTIPO	PROGRAMA EN VISUAL BASIC 6.0
Vista frontal de la puerta en donde se encuentra el sensor magnético 2  Vista interior de la puerta en donde se encuentra el sensor magnético 2 	

Tabla 4.6: Monitoreo del sensor magnético 1.

CAPÍTULO V

5.1. CONCLUSIONES

- Es de suma importancia elegir el PIC correcto tomando en cuenta las características necesarias para el funcionamiento del proyecto entre ellas memoria, número de entradas y salidas, comunicación USART.

- El programa desarrollado en Visual Basic 6.0 va a ser la interfaz con que el usuario va a poder controlar el prototipo.

- La característica principal de este prototipo es la de monitorear y controlar la casa por medio de la PC (luminarias, activación de la alarma, programación del encendido y apagado de las luminarias) y a la vez visualizar que dispositivo esta activado de una forma gráficamente amigable.

- Una de las ventajas del prototipo es que nos ahorra el tiempo que se demora en encender o apagar una luminaria con el interruptor, ya que con hacer un clic en los botones (H1, H2, H3, H4, H5, H6, H7, H8) del programa obtendremos los mismos resultados.

- Al programar la hora de encendido y apagado de luces en la parte externa de nuestro prototipo nos ahorramos tiempo y gastos innecesarios de energía eléctrica.

- Por medio del programa se facilita la verificación del activado de los sensores ya que lo podemos hacer de forma visual a través del programa.

- El sistema de alarma está diseñado de tal manera que monitorea todos los sensores y deja de monitorear el sensor de presencia y los dos sensores magnéticos cuando se desactiva la alarma.

- El sensor de humo y el sensor infrarrojo no dependen de la activación del sistema de alarma, ya que por seguridad el sensor de humo funcionan permanentemente y la función de sensor infrarrojo es el de activar la luminaria de la escalera cuando exista presencia.

- Se utiliza dos reguladores de voltaje 7805 y 7812 y un adaptador de 30V, para el funcionamiento de los dispositivos utilizados.

- Este proyecto será donado al laboratorio de microcontroladores de nuestra prestigiosa Escuela de Formación Tecnológica, para que los estudiantes puedan programar su microcontrolador y puedan verificar con resultados físicos en el prototipo.

5.2. RECOMENDACIONES

- Antes de ser usado el proyecto es recomendable leer el manual de usuario que se encuentra en Anexo F.

- No topar los pines del microcontrolador con las manos ya que estos son delicados y pueden torcerse y al colocarlos en el zócalo pueden romperse los pines.

- Es recomendable que en la etapa de control utilizar borneras para tener mayor facilidad a la conexión y desconexión de la carga.

- Se recomienda asegurarse que los cables conectados a las borneras estén haciendo contacto, caso contrario no funcionará el dispositivo.

- Tenemos que tener en cuenta que a la entrada de la placa existe un fusible que sirve como protección de posibles sobrecorrientes, en caso que se haya quemado el fusible se debe reemplazarlo inmediatamente ya que sin él no funcionaria el prototipo.

- El uso del cable usb serial es la vía de comunicación entre la casa y el programa diseñado en Visual Basic, sin él no se podría realizar el proceso de transmisión y recepción de datos a la PC.

- Cuando se active el sensor de humo, no olvidar de apagar el interruptor del sensor de humo, ya que si no se apaga se mantendrá encendido el sensor.

- Antes de conectar cualquier dispositivo se recomienda verificar a que bornera pertenece, ya que todas las borneras no tiene el mismo voltaje de salida y en el caso de conectar mal algún dispositivo puede darse un daño en el mismo.

- El sistema de alimentación interrumpida o mejor conocido como ups, es recomendable conectar la computadora a un ups, para proporcionar energía eléctrica tras un apagón eléctrico, además otra de las funciones de los UPS es la de mejorar la calidad de la energía eléctrica que llega a las cargas, filtrando subidas y bajadas de tensión de la red. La función del ups es proteger el computador y el sistema en caso de un corte de energía.

- El detector de presencia que se implementó tiene gran inmunidad a falsas alarmas provocados por mascotas, con la posición del jumper se puede discriminar mascotas con peso inferior de 15 kg o 25 kg. (Ver anexo D)

- El uso de la palabra prototipo va dirigido a que el prototipo es un producto terminado al que se llega en la fase de investigación y desarrollo, pero que no es, todavía, el definitivo que se comercializa. Este prototipo se le puede hacer muchas mejoras en cuanto a la seguridad si se desea comercializar,

pero más bien está enfocado al aprendizaje de los estudiantes para que puedan verificar los resultados de la programación en el microcontrolador.

- Lo más recomendable es disponer de un detector de humo por cada 60 metros cuadrados. El lugar óptimo para instalarlo es el centro del techo, ya que en las esquinas el aire no se mueve, también hay que tener la precaución de ubicarlo, como mínimo, a 30 centímetros de cualquier artículo de decoración (lámparas, cuadros) que pueda obstruir la entrada de humo, a si mismo se debe de evitar su ubicación cerca de algunos electrodomésticos, salidas de aire acondicionado y salidas de aire caliente, que pueden desviar el humo fuera del alcance de los detectores.

- Se recomienda usar en la cocina los detectores termovelocímetros ya que estos controlan que la temperatura no aumente a gran velocidad y hacen saltar la alarma cuando ésta aumenta a más de 8°C por minuto.

- Es recomendable usar el sistema de seguridad de una forma adecuada, es decir, activar el sistema cuando se salga del domicilio y desactivarlo al ingresar al domicilio, también considerando pequeñas recomendaciones para la seguridad misma del domicilio como por ejemplo, cerrar la puerta con llave cada vez que entre o salga de su domicilio, no tener animales domésticos dentro de la casa porque el 70% de las falsas activaciones son ocasionadas por éstos, no dejar cosas de valor a la vista sería mucho mejor si se tiene la posibilidad de tener una caja de seguridad para guardar las cosas de mayor valor, en caso de la pérdida de las llaves de su casa lo más aconsejable es reemplazar al menos las cerraduras de las puertas que dan al exterior, todo esto con el fin de precautelar la seguridad de su domicilio.

REFERENCIAS BIBLIOGRÁFICAS

RESISTENCIA ELÉCTRICA

http://www.asifunciona.com/electrotecnia/ke_resistencia/ke_resistencia_1.htm

CONDENSADORES

http://www.planetaelectronico.com/cursillo/tema2/tema2.3.html

TRANSISTOR

http://www.alegsa.com.ar/Dic/transistor.php

http://www.monografias.com/trabajos-pdf/transistores-aplicaciones/transistores-aplicaciones.pdf

Albert Paul Malvino, Principios de Electronica, Sexta edición, Mc Graw Hill, España, 1999.

Robert L. Boylestad, Louis Nashelsky, Electronica: Teoria de Circuitos,sexta Edicion, Pearson Educacion, Mexico, 1997.

OPTOACOPLADORES

http://www.elprisma.com/apuntes/ingenieria_electrica_y_electronica/electronicadepotencia/default5.asp

http://moc3021.blogspot.com/triac

MICROCONTROLADOR

Jose M Angulo Usategui, Ignacio Angulo Martinez,Microcontroladores PIC, Tercera Edicion, Mc Graw Hill. España, 2005.

Carlos A. Reyes, Microcontroladores Pic Programacion en Basic, Segunda Edicion, Rispergraf, Ecuador, 2006.

http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

TRIAC

http://www.monografias.com/trabajos14/triac/triac.shtml

http://www.inele.ufro.cl/bmonteci/semic/applets/pag_triac/triac.htm

COMUNICACION SERIAL

http://www.i-micro.com/pdf/articulos/rs-232.pdf

http://perso.wanadoo.es/pictob/comserie.htm

http://www.iearobotics.com/proyectos/cuadernos/ct1/ct1.html

MAX 232

http://robots-argentina.com.ar/Comunicacion_max232.htm

http://www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf

VISUAL BASIC 6.0

http://www.scribd.com/doc/11493749/Libro-de-ORO-de-Visual-Basic-60-Orientado-a-Bases-de-Datos-2da-Ed

MIKROBASIC

Traducción de la ayuda de Mikrobasic 7.0

ANEXOS

ANEXO A: DATASHEET 78XX (7805 y 7812)

ANEXO B: DATASHEET PIC16F887A

ANEXO C: SENSOR DE HUMO

ANEXO D: SENSOR DE PRESENCIA

ANEXO E: DATASHEET DEL SENSOR INFRAROJO

ANEXO F: MANUAL DE USUARIO

ANEXO A

DATASHEET 78XX (7805 y 7812)

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

KA78XX/KA78XXA

3-Terminal 1A Positive Voltage Regulator

Features

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

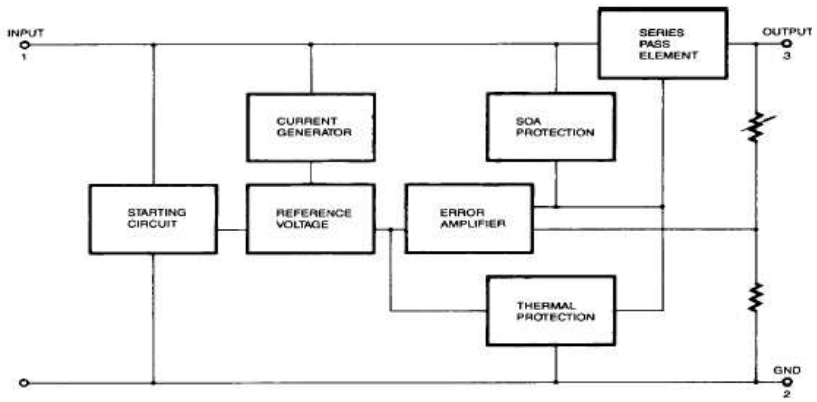
The KA78XX/KA78XXA series of three-terminal positive regulator are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

TO-220

D-PAK

1. Input 2. GND 3. Output

Internal Block Digram



Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	°C/W
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	°C/W
Operating Temperature Range (KA78XX/A/R)	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics (KA7805/KA7805R)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I =10V, C_I= 0.33µF, C_O=0.1µF, unless otherwise specified)

Parameter	Symbol	Conditions		KA7805			Unit
				Min.	Typ.	Max.	
Output Voltage	VO	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	IQ	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔIQ	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	ΔVO/ΔT	IO= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	VN	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	µV/VO
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	VDrop	IO = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	rO	f = 1KHz		-	15	-	mΩ
Short Circuit Current	ISC	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	IPK	T_J =+25 °C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.

Electrical Characteristics (KA7812/KA7812R)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I =19V, C_I= 0.33µF, C_O=0.1µF, unless otherwise specified)

Parameter	Symbol	Conditions		KA7812/KA7812R			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 °C		11.5	12	12.5	V
		5.0mA ≤ I_O≤1.0A, P_O≤15W V_I = 14.5V to 27V		11.4	12	12.6	
Line Regulation (Note1)	Regline	T_J =+25 °C	V_I = 14.5V to 30V	-	10	240	mV
			V_I = 16V to 22V	-	3.0	120	
Load Regulation (Note1)	Regload	T_J =+25 °C	I_O = 5mA to 1.5A	-	11	240	mV
			I_O = 250mA to 750mA	-	5.0	120	
Quiescent Current	I_Q	T_J =+25 °C		-	5.1	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.1	0.5	mA
		V_I = 14.5V to 30V		-	0.5	1.0	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O = 5mA		-	-1	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A =+25 °C		-	76	-	µV/Vo
Ripple Rejection	RR	f = 120Hz V_I = 15V to 25V		55	71	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J=+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	18	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A=+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J = +25 °C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.

Typical Perfomance Characteristics

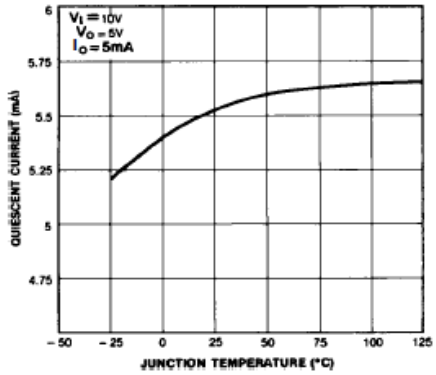


Figure 1. Quiescent Current

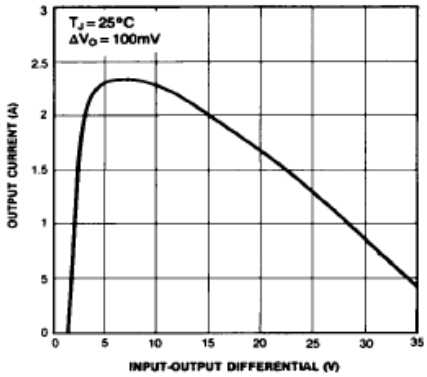


Figure 2. Peak Output Current

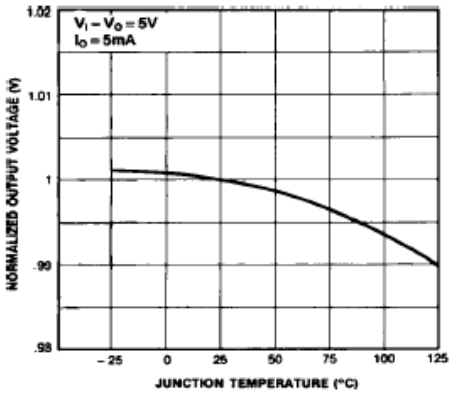


Figure 3. Output Voltage

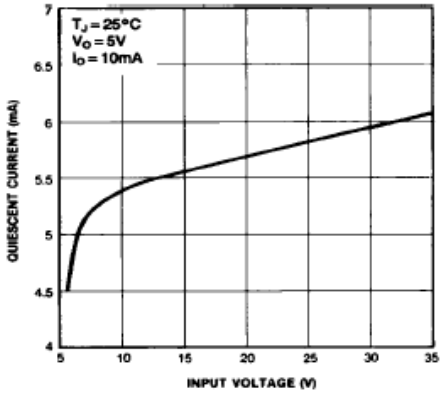


Figure 4. Quiescent Current

Typical Applications

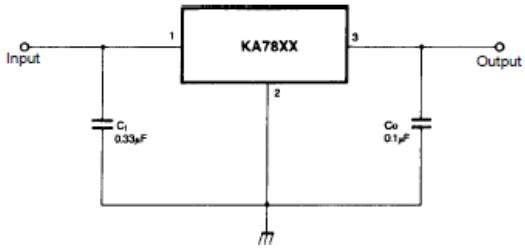


Figure 5. DC Parameters

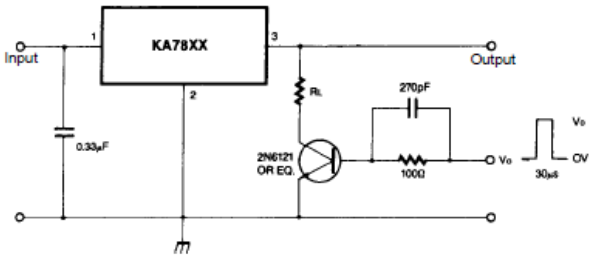


Figure 6. Load Regulation

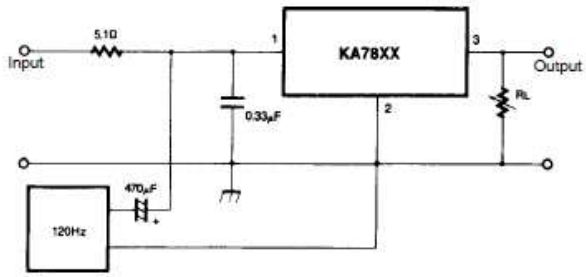


Figure 7. Ripple Rejection

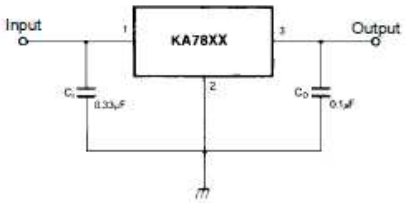


Figure 8. Fixed Output Regulator

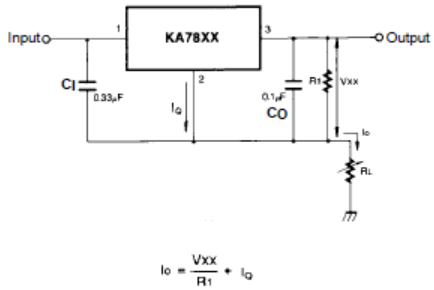


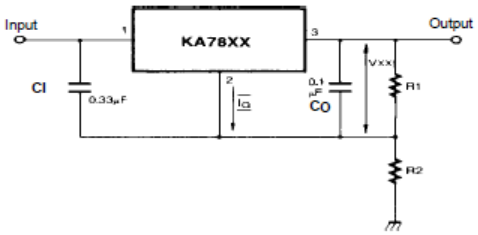
Figure 9. Constant Current Regulator

Notes:

(1) To specify an output voltage. substitute voltage value for "XX." A common ground is required between the input and the Output voltage. The input voltage must remain typically 2.0V above the output voltage even during the low point on the input ripple voltage.

(2) C_I is required if regulator is located an appreciable distance from power Supply filter.

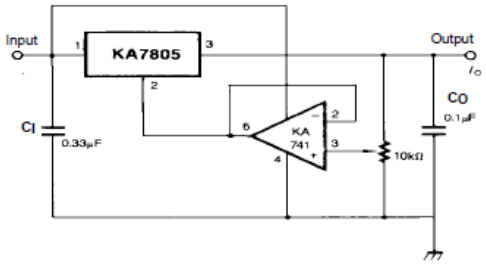
(3) C_O improves stability and transient response.



$I_{RI} \geq 5IQ$

$V_O = V_{XX}(1+R_2/R_1)+I_QR_2$

Figure 10. Circuit for Increasing Output Voltage



$I_{RI} \geq 5\ I_Q$

$V_O = V_{XX}(1+R_2/R_1)+I_QR_2$

Figure 11. Adjustable Output Regulator (7 to 30V)

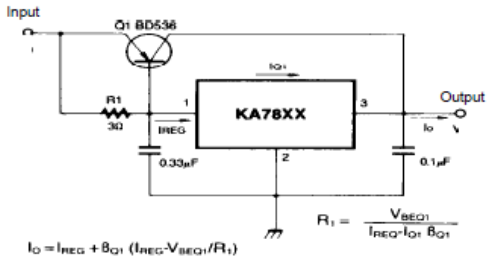


Figure 12. High Current Voltage Regulator

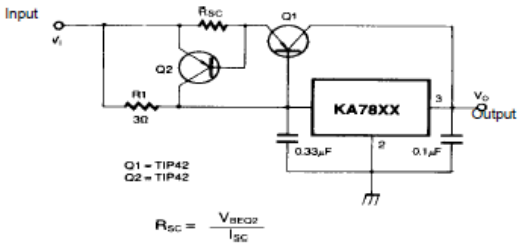


Figure 13. High Output Current with Short Circuit Protection

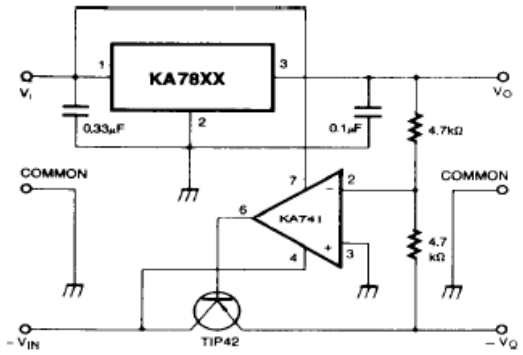


Figure 14. Tracking Voltage Regulator

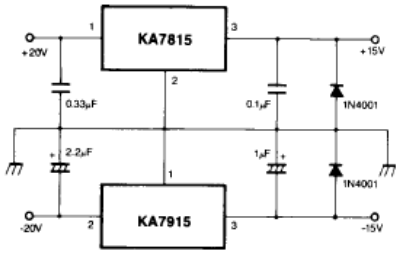


Figure 15. Split Power Supply (±15V-1A)

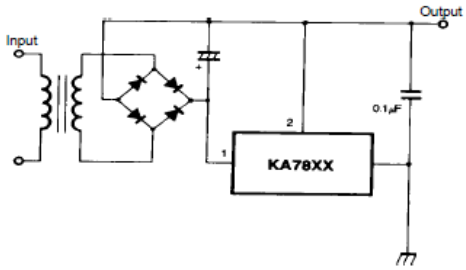


Figure 16. Negative Output Voltage Circuit

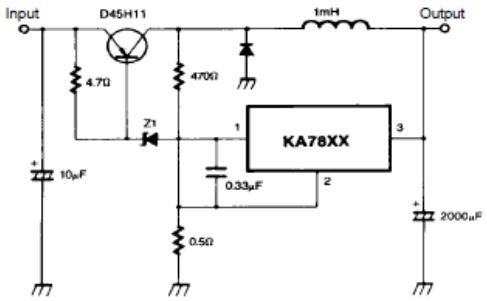


Figure 17. Switching Regulator

ANEXO B

DATASHEET PIC16F887A

MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

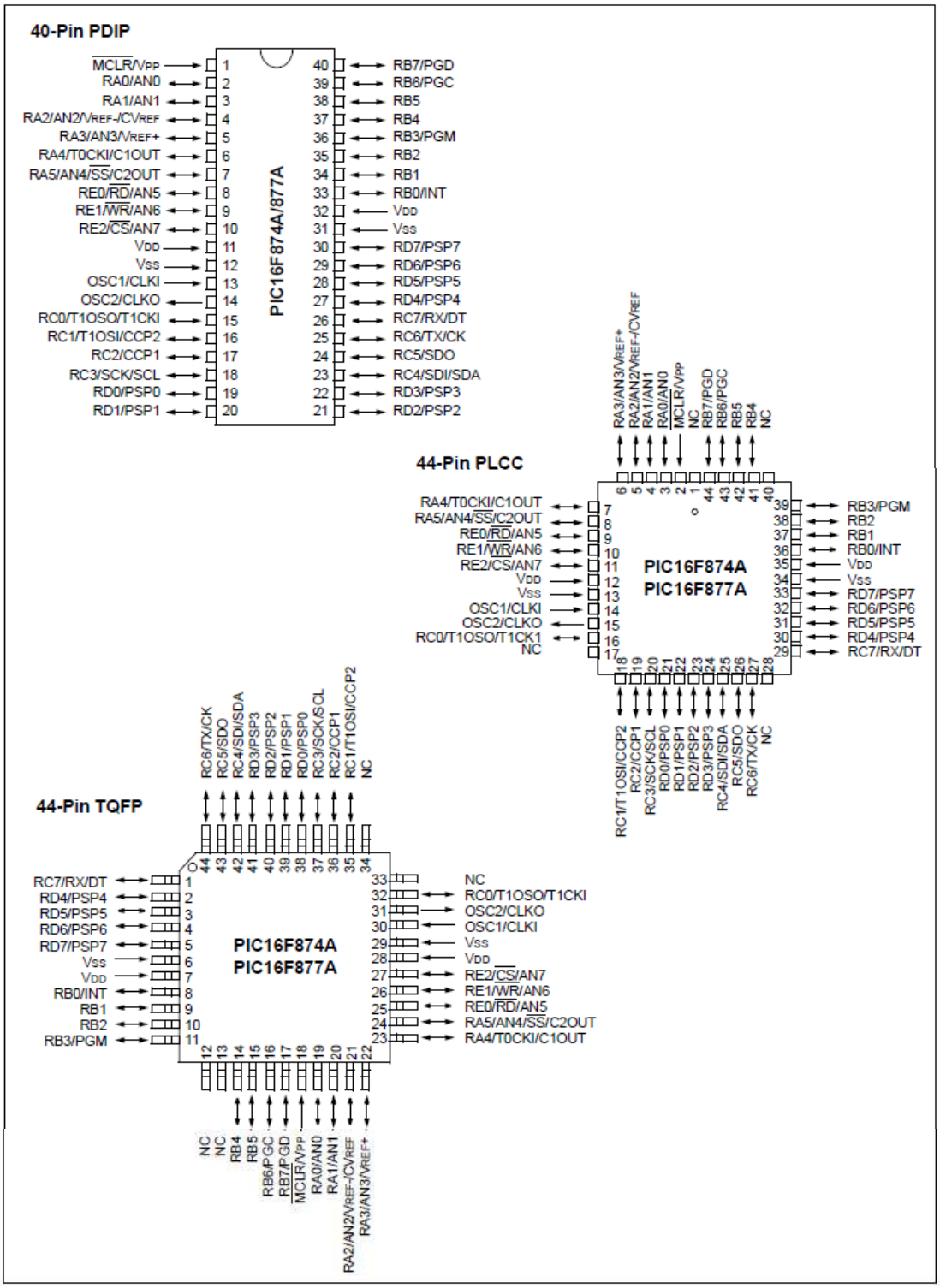
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Pin Diagrams (Continued)



1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implemented only on the 40/44-pin devices

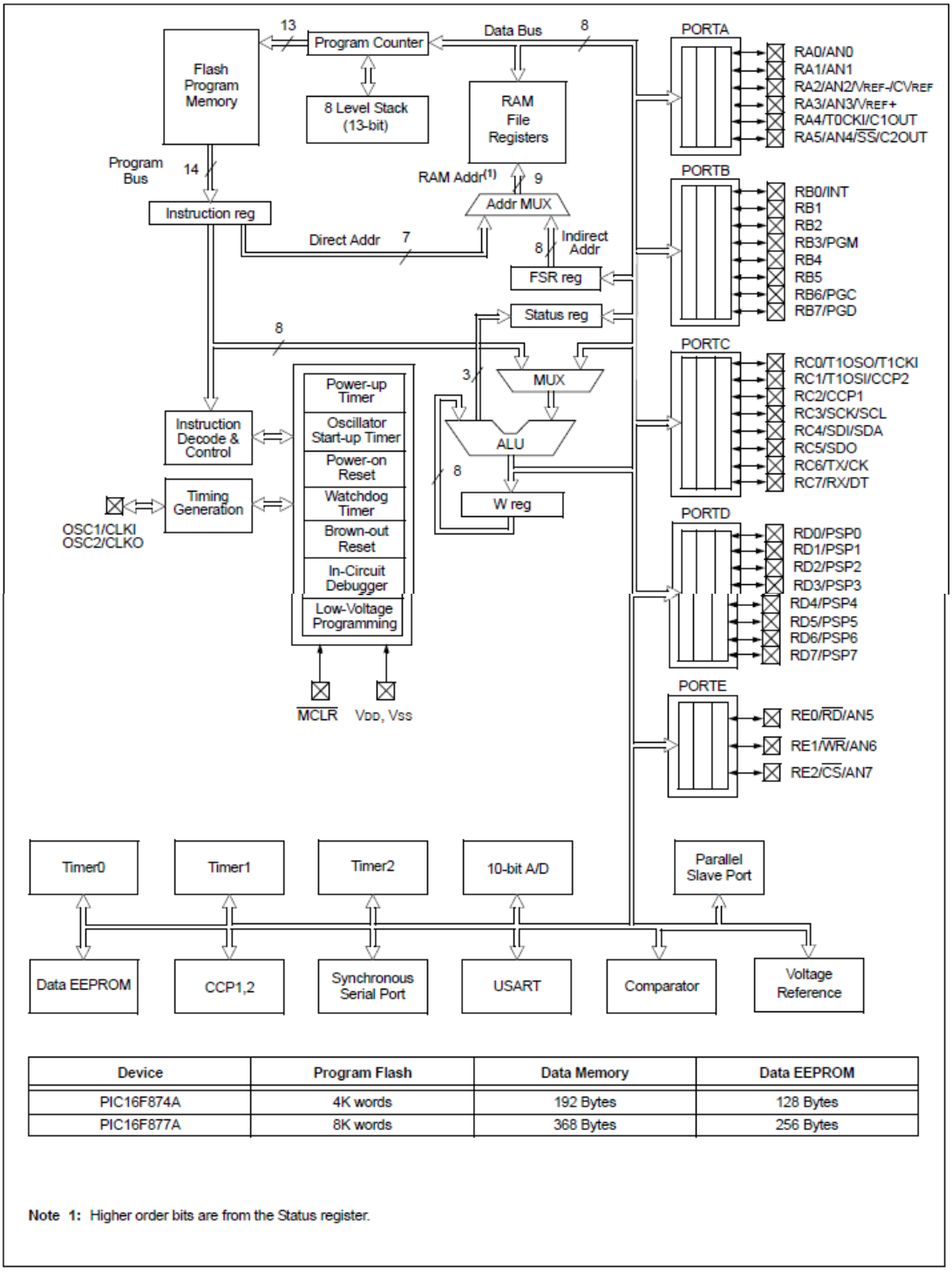
The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

FIGURE 1-2: PIC16F874A/877A BLOCK DIAGRAM



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the Status register.

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
	7Fh		EFh		16Fh		1EFh
accesses 70h-7Fh		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
			FFh		17Fh		1FFh

Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.

2: These registers are reserved; maintain these registers clear.

ANEXO C

SENSOR DE HUMO

Anexo C1: Sensor de humo

ANEXO D

SENSOR DE PRESENCIA

INSTALLATION INSTRUCTIONS

SWAN QUAD

PIR MOTION DETECTOR With PET IMMUNITY up to 25 kg

PRODUCT FEATURES

The SWAN QUAD detector uses a special designed optical Lens with unique Quad (Four element) PIR Sensor and new ASIC based electronics optimized to eliminate false alarms, caused by small animals and Pets.
The SWAN QUAD provides unprecedented levels of immunity against visible light.
The Detector offers an exceptional level of detection capability and stability for every security installation.
The SWAN QUAD is supplied with a Wide Angle lens.

- Quad Linear Imaging Technology for sharp analysis of body dimensions and differentiation from background and animals.
- ASIC based electronics.
- Immunity to animals up to 25kg.
- 18m Detection Range with Wide Angle Lens.
- Temperature compensation.
- Compact Design for Residential Installation.
- Variable pulse width adjustment.
- Sensitivity adjustment.
- Environmental immunity.
- Height installation calibration free (1.8m – 2.4m).
- LED Remote function.

SELECT MOUNTING LOCATION

Choose a location most likely to intercept an intruder. (Our recommendation is a corner installation). See detection pattern fig.3. The quad-element high quality sensor detects motion crossing the beam; it is slightly less sensitive detecting motion toward the detector. The SWAN quad performs best when provided with a constant and stable environment and background.
AVOID THE FOLLOWING LOCATIONS

- Facing direct sunlight.
- Facing areas that may change temperature rapidly.
- Areas where there are air ducts or substantial airflows.

WIRE SIZE REQUIREMENTS

Use #22 AWG (0.5 mm) or wires with a larger diameter. Use the following table to determine the required wire gauge (diameter) depending on the length of wire between the detector and the control panel.

Wire Length	m	200	300	400	800
Wire Diameter	mm	.5	.75	1.0	1.5
Wire Length	ft.	800	1200	2000	3400
Wire Gauge	#	22	20	18	16

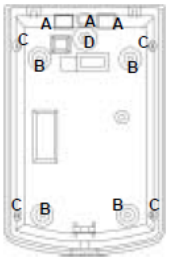
DETECTOR INSTALLATION

The detector can either be wall or corner mounted. If ceiling or special wall mounting is required, use the optional bracket base. Refer to bracket description. (See fig. 6).

1. To remove the front cover, unscrew the holding screw and gently raise the front cover.

Fig.1

2. To remove the PC board, carefully unscrew the holding screw located on the PC board.
3. Break out the desired holes for proper installation.

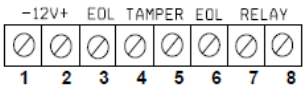


A. Wire access holes
B. Use for flat wall mounting
C. Corner mounting - use all 4 holes. Sharp left or right angle mounting - use 2 holes (top and bottom)
D. For bracket mounting

Fig.2

4. The circular and rectangular indentations at the bottom base are the knockout holes for wire entry. You may also use mounting holes that are not in use for running the wiring into the detector. (For Bracket option - lead wire through the bracket)
5. Mount the detector base to the wall, corner or ceiling. (For bracket installation option see fig. 6).
6. Reinstall the PC board by fully tightening the holding screw. Connect wire to terminal block.
7. Replace the cover by inserting it back in the appropriate closing pins and screw in the holding screw.

DETECTOR CONNECTION



Terminal 1 - Marked " - " (GND)
Connect to the negative Voltage Supply or ground of the control panel.

Terminal 2 - Marked " + " (+12V)
Connect to a positive Voltage Supply of 8.2 -16Vdc source (usually from the alarm control unit)

Terminals 3 & 6 - Marked " EOL " – End of line option.

Terminals 4 & 5 - Marked " TAMPER "
If a Tamper function is required connect these terminals to a 24-hour normally closed protective zone in the control unit. If the front cover of the detector is opened, an immediate alarm signal will be sent to the control unit.

Terminals 7 & 8 - Marked " RELAY "
These are the output relay contacts of the detector. Connect to a normally closed zone in the control panel.

TESTING THE DETECTOR

Wait one minute after applying 12 Vdc power for warm up time. Conduct testing with the protected area cleared of all people.

Walk test

1. Remove front cover.
2. Set LED to ON position.
3. Reassemble the front cover.
4. Start walking slowly across the detection zone.
5. Observe that the LED lights whenever motion is detected.
6. Allow 5 sec. between each test for the detector to stabilize.
7. After the walk test is completed, you can set the LED to OFF position.

NOTE:
Walk tests should be conducted, at least once a year, to confirm proper operation and coverage of the detector.

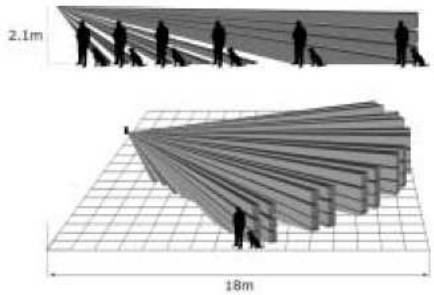


Fig.3

Fig.4

INSTALLATION INSTRUCTIONS

SWAN QUAD

PIR MOTION DETECTOR With PET IMMUNITY up to 25 kg

SETTING UP THE DETECTOR

PET IMMUNITYJUMPER SETTING

This jumper is used for setting the PET Immune function - up to 15kg or 25kg, depending on the pet size.

Immune to pet weighting up to 15 kg

Immune to pet weighting up to 25 kg

PULSE WIDTH JUMPER SETTING

This jumper is used for setting the PULSE count function in order to provide PIR sensitivity control according to the environment.

Very stable environment
Jumper #1 = ON
Without PET

Moderate nuisance situation
Jumper #2 = ON
PET up to 15 kg

Relatively high chance of false alarms
Jumper #3 = ON
PET up to 25 kg

LED ENABLE JUMPER SETTING

This jumper is used for setting - LED Enable / Disable.

ON - LED ENABLE. The LED will activate when the detector is in alarm condition.

OFF - LED DISABLE, The LED is disabled.

Note: The LED Switch does not affect the operation of the relay.
When an intrusion is detected, the LED will activate and the alarm relay will switch into alarm condition for 2 sec.

PIR SENSITIVITY ADJUSTMENT

Use the Potentiometer marked "SENS" to adjust the detection sensitivity between 15% and 100%, according to walk test in the protected area.
(Factory set to 57%)
Rotate the potentiometer clockwise to increase range, counter-clockwise to decrease range.

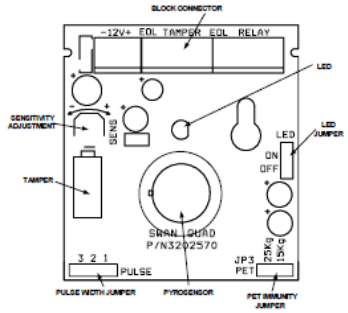


Fig.5

BRACKET INSTALLATION OPTION

Ceiling bracket base | Wall bracket base

Fig. 6

TECHNICAL SPECIFICATION

MODEL	*SWAN QUAD*
Detection Method	*Quad (Four element) PIR*
Power Input	*8.2 to 16 VDC*
Current Draw	*Standby: 8mA (± 5%) Active: 10mA (± 5%)*
Temperature Compensation	*YES*
Pulse Width	*Adjustable*
Alarm Period	*2 sec (± 0.5sec)*
Alarm Output	*N.C 28VDC 0.1 A with 27Ohm series protection resistor*
Tamper Switch	*N.C 28VDC 0.1A with 10 Ohm series protection resistor - open when cover is removed*
Warm Up Period	*60sec (± 5sec)*
LED Indicator	*LED is ON during alarm*
Operating Temperature	*-20℃ to +60℃*
RFI Protection	*30V/m 10 - 1000MHz*
EMI Protection	*50,000V of electrical interference from lightning or power through*
Dimensions	*92mm x 59mm x 37mm*
Weight	*40gr*

N345

CE

CROW ELECTRONIC ENGINEERING LTD. ("Crow") - WARRANTY POLICY CERTIFICATE

This Warranty Certificate is given in favor of the purchaser (hereunder the "**Purchaser**") purchasing the products directly from Crow or from its authorized distributor.

Crow warrants these products to be free from defects in materials and workmanship under normal use and service for a period of 1 year from the last day of the week and year whose numbers are printed on the printed circuit board inside these products (hereunder the "**Warranty Period**").
Subject to the provisions of this Warranty Certificate, during the Warranty Period, Crow undertakes, at its sole discretion and subject to Crow's procedures, as such procedures are form time to time, to repair or replace, free of charge for materials and/or labor, products proved to be defective in materials or workmanship under normal use and service. Repaired products shall be warranted for the remainder of the original Warranty Period.

All transportation costs and in-transit risk of loss or damage related, directly or indirectly, to products returned to Crow for repair or replacement shall be borne solely by the Purchaser.

Crow's warranty under this Warranty Certificate does not cover products that is defective (or shall become defective) due to: (a) alteration of the products (or any part thereof) by anyone other than Crow; (b) accident, abuse, negligence, or improper maintenance; (c) failure caused by a product which Crow did not provide; (d) failure caused by software or hardware which Crow did not provide; (e) use or storage other than in accordance with Crow's specified operating and storage instructions.

There are no warranties, expressed or implied, of merchantability or fitness of the products for a particular purpose or otherwise, which extend beyond the description on the face hereof.
This limited Warranty Certificate is the Purchaser's sole and exclusive remedy against Crow and Crow's sole and exclusive liability toward the Purchaser in connection with the products, including without limitation - for defects or malfunctions of the products. This Warranty Certificate replaces all other warranties and liabilities, whether oral, written, (non-mandatory) statutory, contractual, in tort or otherwise.
In no case shall Crow be liable to anyone for any consequential or incidental damages (inclusive of loss of profit, and whether occasioned by negligence of the Crow or any third party on its behalf) for breach of this or any other warranty, expressed or implied, or upon any other basis of liability whatsoever. Crow does not represent that these products can not be compromised or circumvented; that these products will prevent any person injury or property loss or damage by burglary, robbery, fire or otherwise; or that these products will in all cases provide adequate warning or protection.
Purchaser understands that a properly installed and maintained product may in some cases reduce the risk of burglary, fire, robbery or other events occurring without providing an alarm, but it is not insurance or a guarantee that such will not occur or that there will be no personal injury or property loss or damage as a result.
Consequently, Crow shall have no liability for any personal injury; property damage or any other loss based on claim that these products failed to give any warning.
If Crow is held liable, whether directly or indirectly, for any loss or damage with regards to these products, regardless of cause or origin, Crow's maximum liability shall not in any case exceed the purchase price of these products, which shall be the complete and exclusive remedy against Crow.

CROW ELECTRONIC ENGINEERING LTD.

ISRAEL: 57 Hamelacha St., Holon 58855
Tel: 972-3-5569937 /8 /9
Fax: 972-3-5592981
E-mail: support@crow.co.il

USA: 2160 North Central Road,
Fort Lee, N.J. 07024
Tel: 1-800-GET CROW
or (201) 944 0005
Fax: (201) 944 1199
E-mail: support@crowelec.com

AUSTRALIA: 429 Nepean HWY Brighton East Vic 3187
Tel: 61-3-9596 7222
Fax: 61-3-9596 0888
E-mail: crow@crowaust.com.au

POLAND: VIDICON SP. ZO. O.
15 Powązkowska St.
01 – 797 Warsaw Poland
Tel: 48 22 562 3000
Fax: 48 22 562 3030
E-mail: vidicon@vidicon.pl

LATIN AMERICA: CROW LATIN AMERICA
5753 NW 151st Street
MIAMI LAKES,
FL 33014 – USA
Tel: +1-305-823-8700
Fax: +1-305-823-8711
E-mail: sales@crowlatinamerica.com

ITALY: DEATRONIC
VIA Giuliamello 4/14
00178 ROMA, ITALY
Tel: +39-0676-12912
Fax: +39-0676-12601
E-mail: info@deatronic.com

These instructions supersede all previous issues in circulation prior to October 2004.

ANEXO E

DATASHEET DEL SENSOR INFRAROJO

PARALLAX

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Sharp GP2D12 Analog Distance Sensor (#605-00003)

General Description

The Sharp GP2D12 is an analog distance sensor that uses infrared to detect an object between 10 cm and 80 cm away. The GP2D12 provides a non-linear voltage output in relation to the distance an object is from the sensor and interfaces easily using any analog to digital converter.

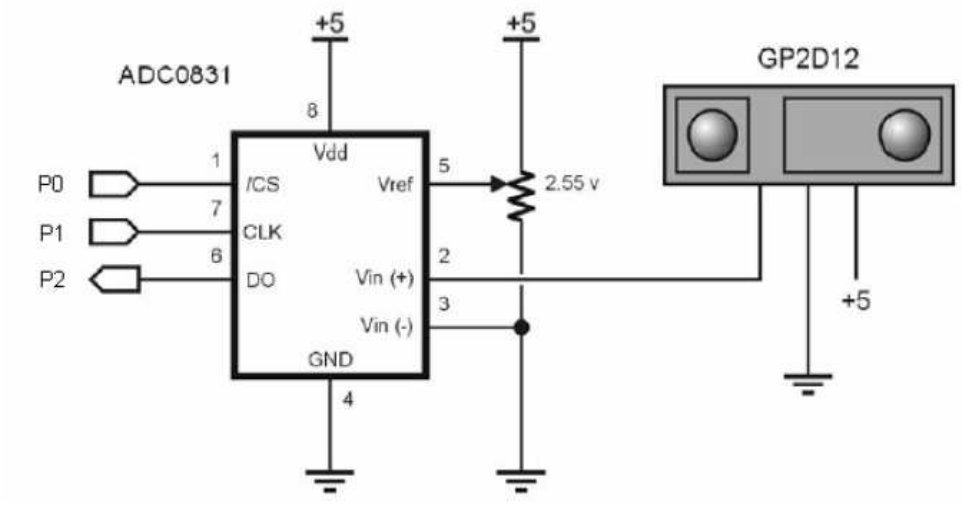
Features

- High immunity to ambient light and color of object
- No external control circuitry required
- Sensor includes convenient mounting holes
- Compatible with all BASIC Stamp® and SX microcontrollers

Application Ideas

- Robot range finder
- Halloween prop activation

Quick Start Circuit

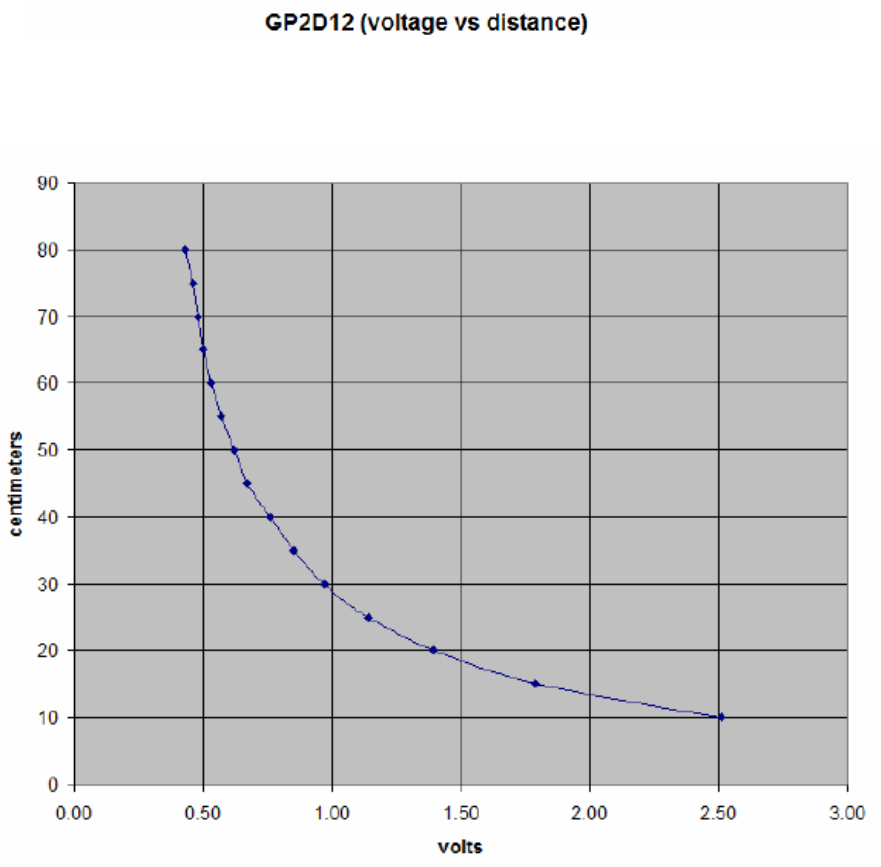


Connecting and Testing

Connect the GP2D12 to your analog to digital converter as shown in the circuit on the previous page. The potentiometer connected to the Vref pin on the ADC0831 is being used as a voltage divider to set the reference voltage to 2.55 volts. On the ADC0831 this will give a value of 0 to 255 for an input voltage of 0 to 2.55 volts. This gives us a resolution of 0.01 volts per step from the ADC. If you are using a different analog to digital converter, you may want to adjust the potentiometer to get the best results from your particular ADC.

Calibration

Because the output of the GP2D12 is not linear, we need a way to determine what distances correspond to what voltages. One way of calibrating your sensor is by measuring the voltage output of the GP2D12 at given fixed distances, in centimeters, as shown in the chart below. Once you have this information you can plug these numbers into the EEPROM DATA statements in the program. The table of data is used by a routine in the program to calculate the distances, which are then displayed on the Debug Terminal, along with the voltage output from the sensor.



Sensitivity

The usable range of the GP2D12 is between 10 cm and 80 cm. The readings for objects closer than 10 cm are unstable and therefore not usable.

Resources and Downloads

Check out the Sharp GP2D12 Analog Distance Sensor product page for example programs, the manufacturer datasheet and more:

http://www.parallax.com/detail.asp?product_id=605-00003

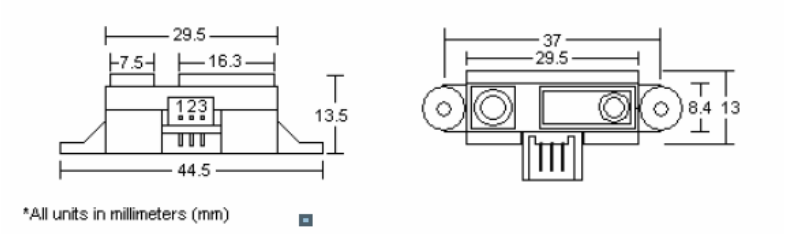
Specifications

Symbol	Quantity	Minimum	Typical	Maximum	Units
Vcc	Supply Voltage †	4.5	5.0	5.5	V
Topr	Operating Temperature †	-10	-	+60	ºC
Tstg	Storage Temperature †	-40	-	+70	ºC
ΔL	Distance Measuring Range †	10	-	80	cm
Vo	Output Terminal Voltage (L=80 cm) †	0.25	0.4	0.55	V
ΔVo	Output change at L=80 cm to 10 cm †	1.75	2.0	2.25	V
Icc	Average Dissipation Current (L=80cm) †	-	33	50	mA

† data obtained from Sharp's GP2D12 datasheet

Pin Definitions and Ratings

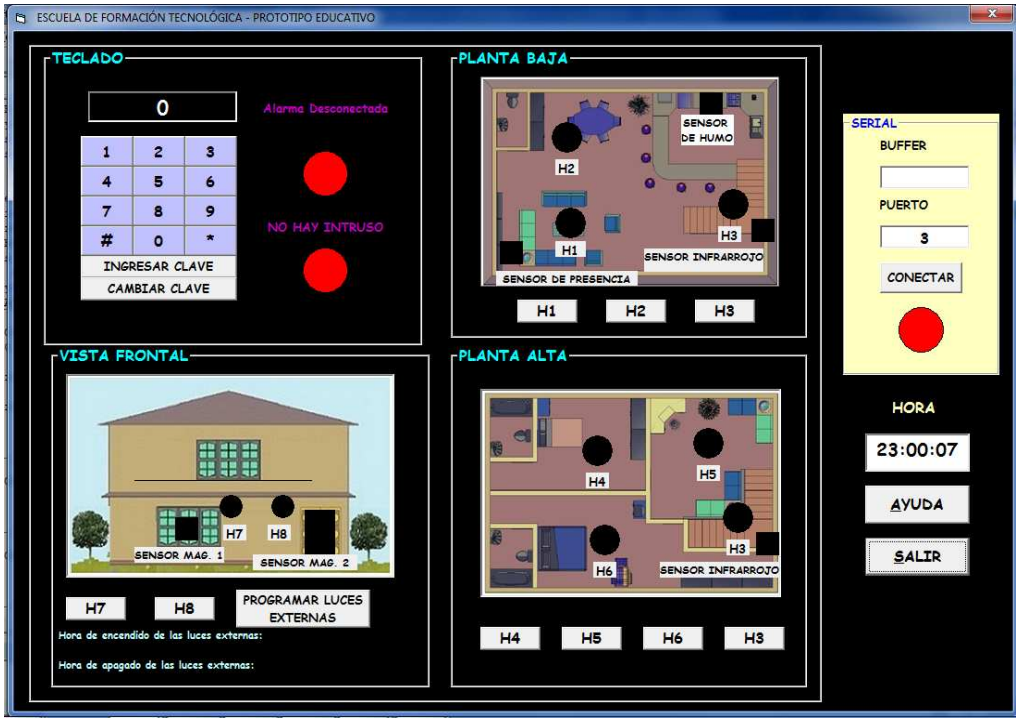
Pin	Name	Function
1	Vo	Voltage Output
2	GND	Ground
3	Vcc	Supply Voltage



ANEXO F

MANUAL DE USUARIO

En Visual Basic tenemos la siguiente interfaz la misma que nos va a permitir monitorear lo que sucede en nuestro prototipo.



A continuación podemos controlar los cambios realizados en las ocho luminarias, un sensor de presencia, un sensor infrarrojo, un sensor de humo y dos sensores magnéticos.

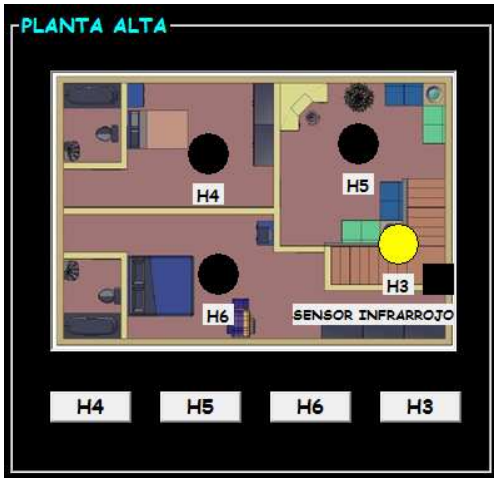
1. LUMINARIAS

En la sección de la VISTA FRONTAL, PLANTA BAJA y PLANTA ALTA DE LA CASA, encontramos las ocho luminarias (H1, H2, H3, H4, H5, H6, H7, H8) que las podemos visualizar en el programa como un círculo de color negro. Las luminarias las podemos encender y apagar por medio de los interruptores o por medio del programa de Visual Basic 6.0, por cualquiera de los dos medios de activación observaremos que la luminaria que estaba representada por el círculo negro cambia al color amarillo.

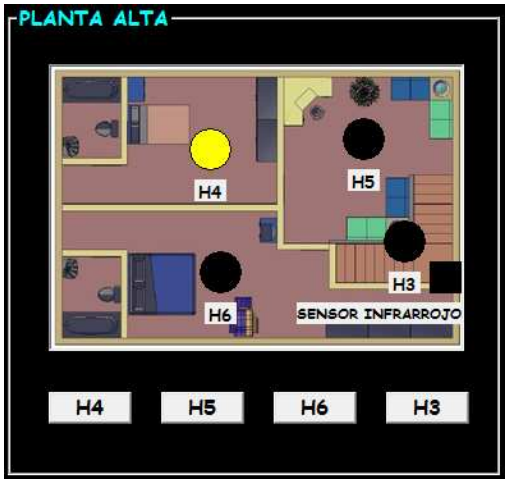
El H1 representa a la luminaria encendida que se encuentra en la sala de la PLANTA BAJA:

El H2 representa a la luminaria encendida que se encuentra en el comedor de la PLANTA BAJA:

El H3 representa a la luminaria encendida que se encuentra en las gradas, la misma que podemos visualizar en las secciones de la PLANTA BAJA y la PLANTA ALTA. Esta luminaria la podemos encender desde el botón H3 o desde el interruptor, pero su función general es que se encienda y se apague la luz automáticamente cuando detecte presencia:



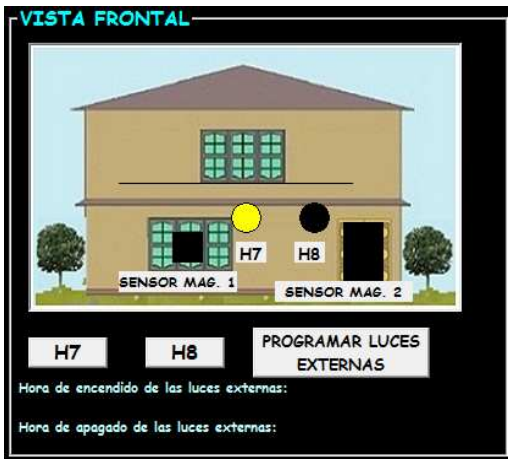
El H4 representa a la luminaria encendida que se encuentra en el dormitorio de la PLANTA ALTA:

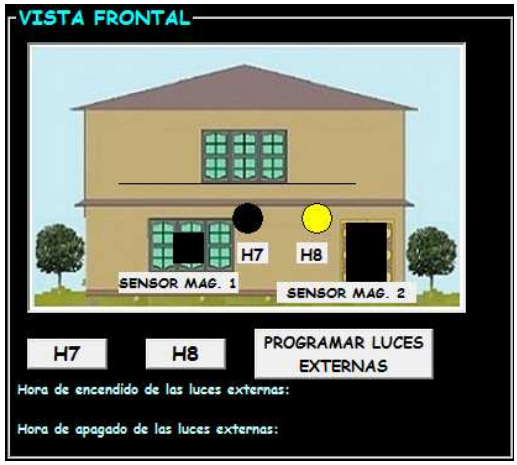


El H5 representa a la luminaria encendida que se encuentra en la sala de estar de la PLANTA ALTA:

El H6 representa a la luminaria encendida que se encuentra en el dormitorio master de la sala de estar de la PLANTA ALTA:

El H7 y el H8 representan a las luminarias externas de la casa que están encendidas y se encuentra en la VISTA FRONTAL de la planta baja de la casa:

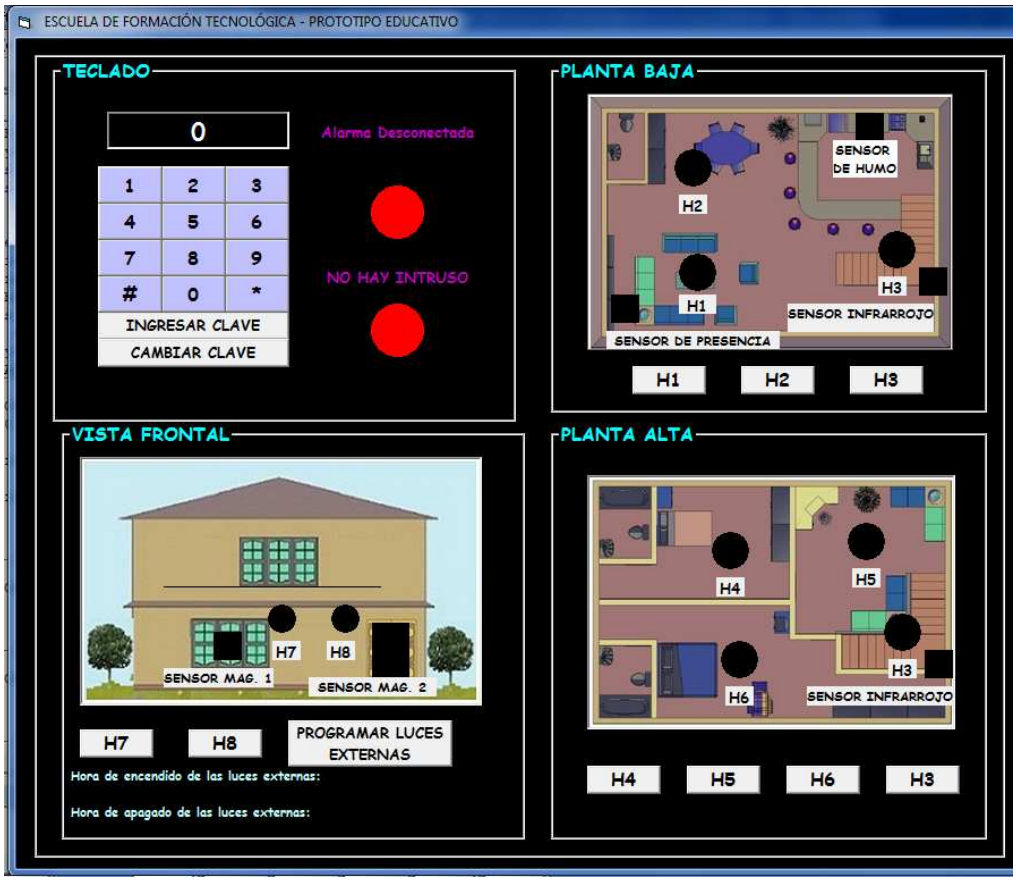




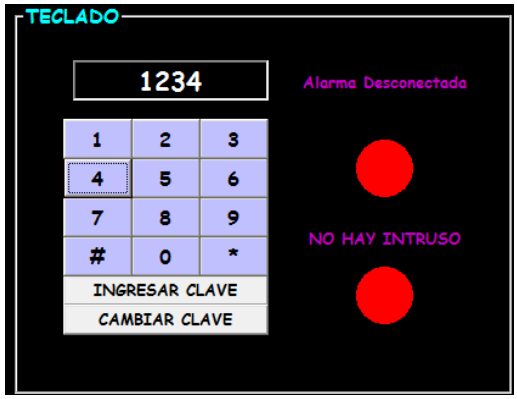
El botón PROGRAMAR LUCES EXTERNAS como su nombre lo indica nos sirve para programar que en un determinado intervalo de tiempo estén encendidas las luminarias.

2. SENSOR DE PRESENCIA Y SENSORES MAGNÉTICOS

Para que entren en funcionamiento el sensor de presencia y los dos sensores magnéticos se debe activar la alarma ingresando la clave por teclado. En la sección del TECLADO encontramos dos mensajes que son de mucha utilidad a lo largo del control del prototipo ya que los dos círculos rojos nos indican que la alarma esta desactivada y que no hay algún intruso en la casa respectivamente, cuando uno de ellos se activa su color cambia de rojo a verde.

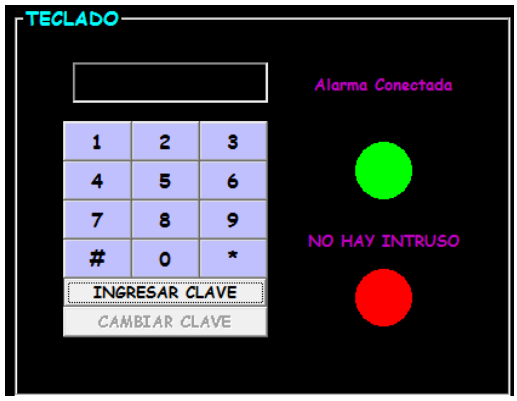


Para activar la alarma ingresamos por medio del teclado nuestra clave que en nuestro caso es 1234 y luego hay que pulsar el botón INGRESAR CLAVE como observamos en la siguiente figura:

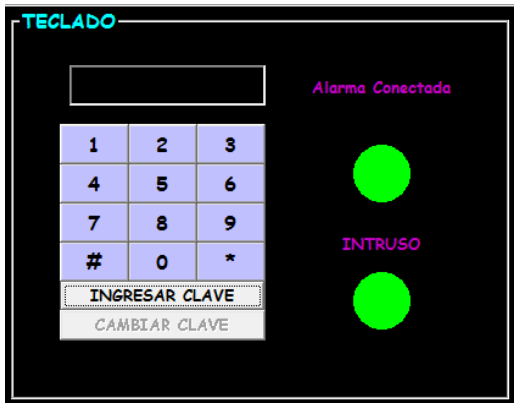


Para la demostración en el prototipo hemos programado un tiempo de cinco segundos para que comience a funcionar la alarma:

Luego de este tiempo el círculo verde nos indica que la alarma está conectada y mientras no haya ningún intruso el círculo rojo permanecerá de ese color:

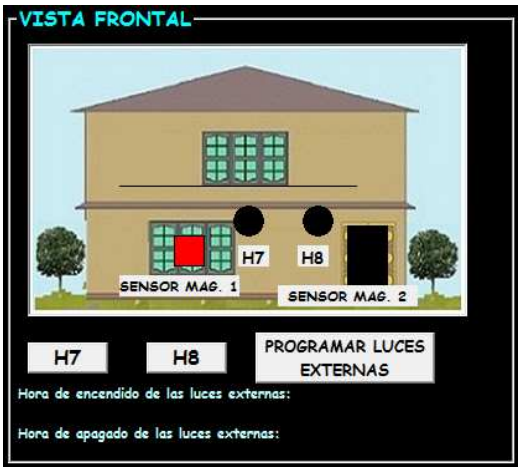


En el momento que exista la presencia de algún intruso en la sección del TECLADO observaremos que las dos advertencias estarán de color verde:

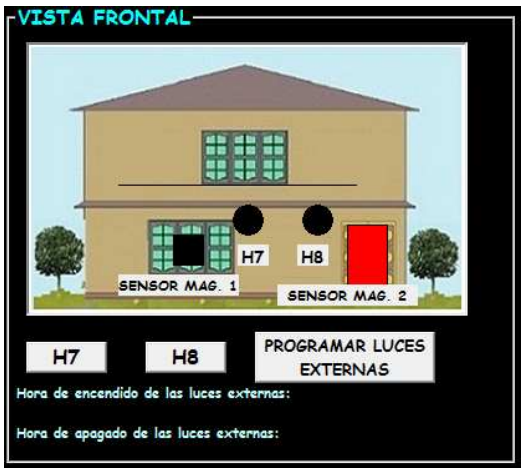


Cuando se activan los sensores magnéticos y el de presencia podemos visualizar que el color negro de la figura que los representa cambia al color rojo, el sensor de presencia se encuentra en la sala de la PLANTA BAJA:

El primer sensor magnético se lo ha colocado en la ventana de la VISTA FRONTAL ubicado en la planta baja de la casa:



El segundo sensor magnético se lo ha colocado en la puerta de la VISTA FRONTAL ubicado en la planta baja de la casa:



3. SENSOR INFRARROJO

En el instante que este sensor se activa visualizamos que el color negro de la figura que los representa cambia al color rojo y que la luminaria H3 se enciende, el color amarillo lo indica. En las dos secciones (PLANTA BAJA y PLANTA ALTA) observamos lo mismo ya que la escalera conecta las dos plantas.

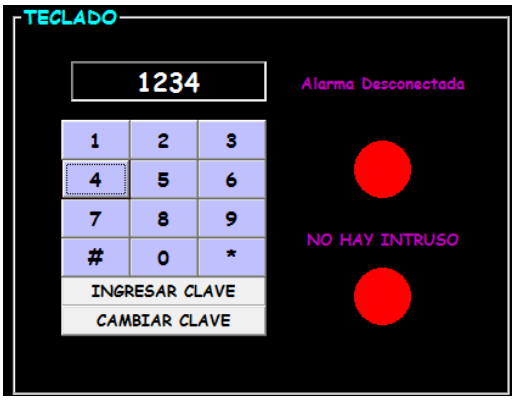
4. SENSOR DE HUMO

El sensor de humo estará en constante funcionamiento, ya sea cuando la alarma este activada o desactivada, en el momento que se active observaremos que el color negro de la figura que los representa cambia al color rojo.

5. CAMBIO DE CLAVE

La clave tiene un máximo de cuatro números y es 1234, si el usuario desea cambiar la clave para la activación de la alarma lo puede hacer solo cuando la alarma de la casa este desconectada. A continuación los pasos:

Dar un clic en el botón CAMBIAR CLAVE:



Automáticamente observamos que nos aparece un mensaje indicándonos que ingresemos la clave actual, dar clic en aceptar y nos aparece otro pantalla en la que vamos a realizar el proceso:

Como nos indica el mensaje ingresamos la clave actual, luego dar clic en el botón INGRESAR CLAVE:

Y nuevamente observaremos otro mensaje que nos indica que ingresemos una nueva clave, damos clic en Aceptar:

Ingresamos la nueva clave que en este ejemplo es 5678 y nuevamente hacemos clic en el botón INGRESAR CLAVE:

Para finalizar este proceso observamos un último mensaje que nos indica la nueva clave.

El botón LIMPIAR nos sirve para borrar si nos equivocamos al ingresar la clave y con el botón SALIR regresamos al formulario inicial.

6. COMINUCACIÓN SERIAL

Para que entre en funcionamiento el programa primeramente debemos ingresar el puerto con que vamos a trabajar, el mismo que pertenece al cable usb serial que usaremos para la comunicación, luego damos un clic en el botón CONECTAR, si se realizó la comunicación correctamente el circulo ubicado debajo del botón CONECTAR cambia del color rojo que esta por defecto al color verde:

7. HORA, AYUDA, FIN

El programa también nos ofrece la visualización de un reloj que nos indica la hora actual, el botón AYUDA tiene como objetivo intentar resolver las dudas que se planteen sobre el manejo del programa de control de Visual Basic y el botón SALIR nos permite cerrar todo el sistema.