TECNOLÓGICO
DE MONTERREY®

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México
División de Ingeniería y Arquitectura

Ingeniería en Sistemas Electrónicos

Departamento de Ingeniería Eléctrica y Electrónica

"Interfaz para personas Paraplejicas"

Autores: Alejandra González Romero 954821
Alfredo Baltazar Rodríguez 717594
Carlos M. González Flores 718271

INDICE

INTRODUCCIÓN

Dentro del campo de la ingeniería podemos encontrar a la biomédica, que se basa en el uso de la física, las matemáticas, en general de la tecnología para la creación de sistemas o aparatos que solucionen ciertos problemas que surgen día con día, más específicamente en la medicina. Siendo diversas las áreas involucradas, la ingeniería biomédica es una actividad interdisciplinaria y multiprofesional, que contribuye tanto al desarrollo científico, económico y social, como al bienestar en general. La tecnología biomédica es actualmente una de las piezas clave de los sistemas de salud, teniendo implicaciones importantes en el costo y la calidad de los servicios. Es por esto que las organizaciones de salud están interesadas en fórmulas que les permitan mejorar los servicios y en lo posible reducir los costos.

Así, se considera como parte de su estudio:

- El desarrollo de tecnología biomédica, nuevos sistemas, dispositivos, procesos y algoritmos, en servicios de la salud.

- Creación de mejores condiciones en los recursos tecnológicos para la prestación de los servicios de salud con calidad.

- El entendimiento y la utilización del conocimiento de los sistemas vivos para aplicaciones clínicas sustantivas e innovadoras basados en las ciencias de la ingeniería.

A nivel de las Instituciones Prestadoras de Servicios de Salud (IPS) se requiere de una planeación en tecnología mediante un proceso racional de adquisición y utilización que beneficie a éstas, a los usuarios y al sistema en general. En este sentido se propone adoptar metodologías de adquisición, prevenir el desbordamiento de los costos que puede producir la compra indiscriminada de alta tecnología y realizar esfuerzos regionales y nacionales en este sentido.

La experiencia internacional, especialmente en los sistemas competitivos, demuestra que los hospitales compiten entre ellos adquiriendo alta tecnología debido a que los pacientes a menudo consideran a la tecnología de alto costo y de punta como señal de calidad.

Para desarrollar este proyecto nos enfocamos en las personas que presentan una lesión en la médula espinal, más específicamente en la triplejia y paraplejia.

Las personas con paraplejia y triplejia tienen una pérdida de sensibilidad e incluso parálisis en las extremidades. En el caso de la paraplejia, la parálisis se da en las extremidades inferiores. Mientras que en la triplejia, las zonas afectadas son las piernas y un brazo.

En la figura 1 podemos observar las diferencias entre la paraplejia y la triplejia.

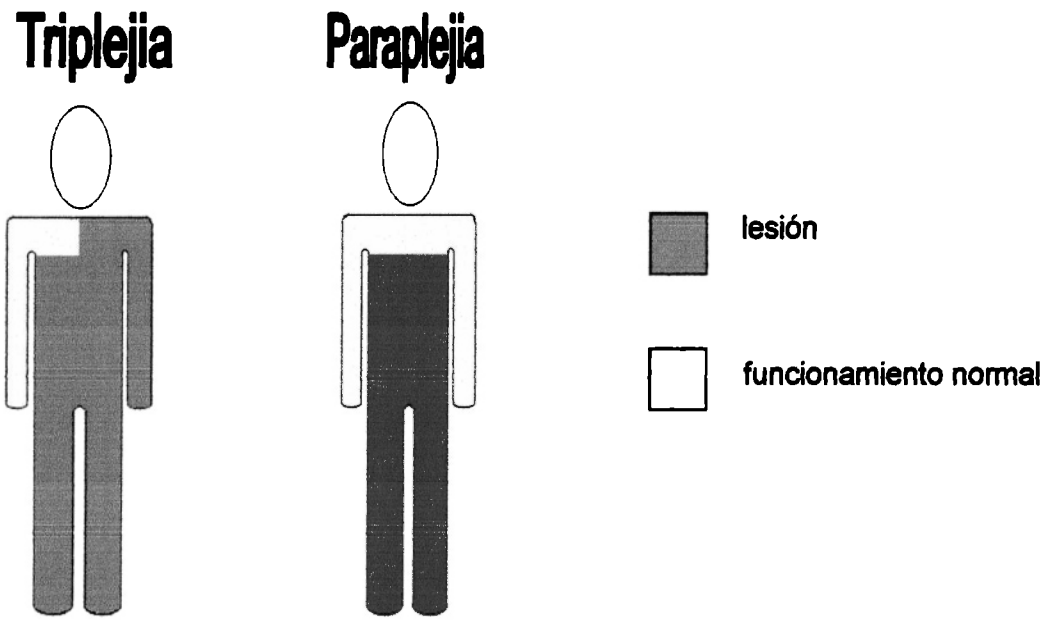


Figura 1 Comparación entre paraplejia y triplejia

En general las lesiones más frecuentes se producen cuando una zona de la columna vertebral o del cuello se dobla o se comprime; por ejemplo, como consecuencia de:

- Lesiones durante el parto, que suelen afectar a la columna vertebral en la zona del cuello.
- Caídas.
- Accidentes de tráfico.
- Lesiones deportivas (principalmente clavadistas).
- Las heridas penetrantes que atraviesan la médula espinal, como por ejemplo los disparos o puñaladas, también pueden causar daños.

Objetivos

- Elaborar una herramienta que permita a las personas con paraplejia tener cierto nivel de independencia en su vida cotidiana.
- Lograr que la herramienta sea de bajo costo.
- Diseñar la herramienta de tal forma que a cualquier persona le parezca fácil de usar.

Justificación

El total de personas con capacidades diferentes, registradas por el INEGI en el año 2000, fueron 1 795 300, donde el 45.3% poseen algún problema motriz. Estamos hablando de 807 885 ciudadanos Mexicanos.

Basándonos en ese estudio y considerando que el número de personas que padecen parálisis es significativamente grande, deseamos aportar un sistema que ayude a estas personas a valerse por sí mismas gozando de cierta comodidad y que tengan una vida digna y feliz.

Actualmente en el mercado existen muchos aparatos que deberían ser utilizados por las personas que padecen paraplejia o triplejia, pero desgraciadamente son muy sofisticados y con un precio muy elevado, por lo que la mayoría de las veces, dichos aparatos sólo se encuentran en clínicas especializadas y no son de uso común.

Es por esto que surge la idea de desarrollar un aparato sencillo de usar y con un precio económico para que sea útil y funcional entre la gran población de paraplejicos y triplejicos.

Definición del Proyecto

La "Interfaz para personas con paraplejia/triplejia" es una herramienta que combina aspectos tecnológicos, tanto hardware como software, ofreciendo la posibilidad de realizar las siguientes actividades de la vida diaria:

- Encender/apagar la luz de una recamara
- Encender/apagar, cambiar el canal y/o el volumen de una televisión
- Activar una alarma sonora y enviar un mensaje SMS (servicio de mensajes cortos a teléfonos celulares), en cualquier situación de emergencia

Todas estas actividades se realizan por medio de un mouse inalámbrico.

Ventajas y desventajas

Ventajas:

- Es muy sencilla de manejar.
- El peso del mouse es relativamente bajo, por lo que la persona no necesita realizar mucha fuerza física.
- No existe limitante alguna en cuanto a que marcas de televisión se pueden usar (SONY, PANASONIC, HITACHI, etc).
- El precio de la interfaz es bajo en comparación con otros aparatos existentes en el mercado.

Desventajas:

- Se necesita una PC conectada a Internet.
- El mensaje de texto sólo está diseñado para teléfonos que tengan contrato con la empresa "UNEFON".

Funcionamiento

La Interfaz para personas con paraplejia/triplejia trabaja de la siguiente manera:

1. Cuando la persona desea realizar alguna acción de las descritas anteriormente, utiliza el mouse inalámbrico para seleccionar la que desea, interactuando con el programa previamente instalado en una PC (Computadora Personal).

2. Una vez seleccionada la acción, ésta se lleva a cabo por medio de los circuitos conectados a la PC por el puerto paralelo.

DESARROLLO

El proyecto básicamente consta de 2 partes:

- En la primera se engloba el mouse inalámbrico, la interfaz gráfica y el envío de mensajes SMS (software)
- En la segunda todos los circuitos (hardware) que activan la acción requerida.

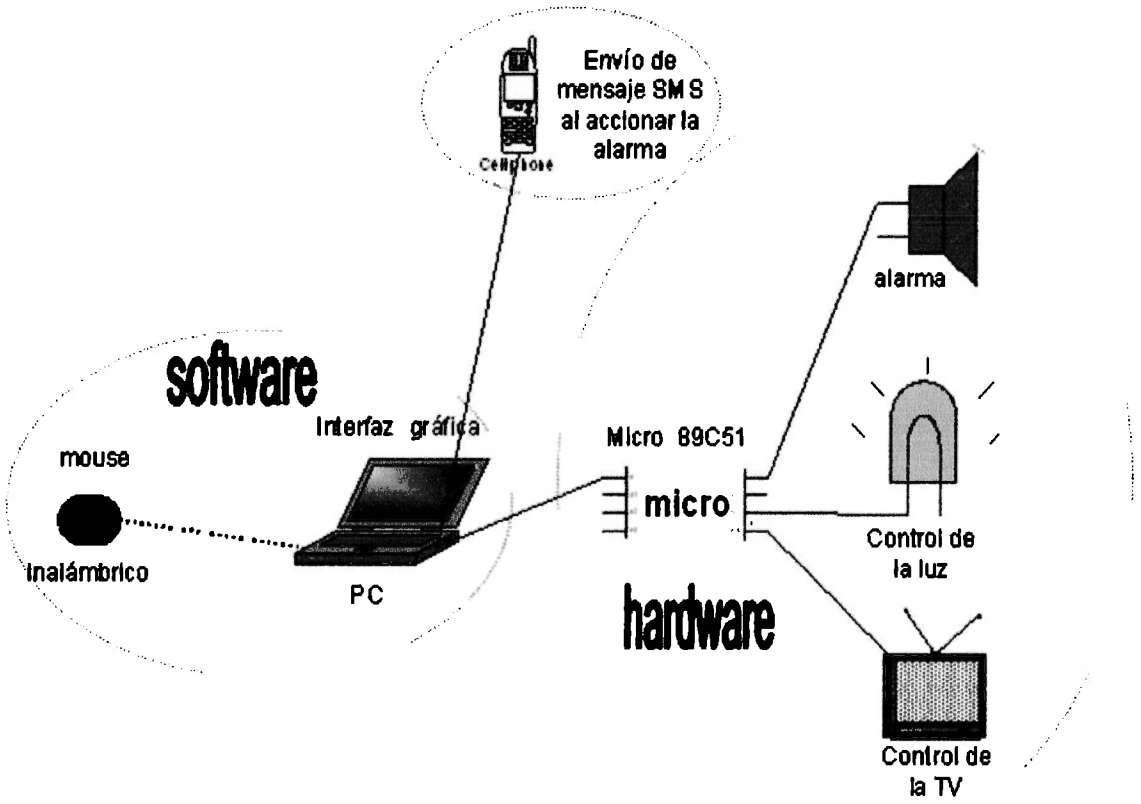


Figura 2 Módulos en los que se divide la interfaz para personas paraplejicas

Módulo 1 Mouse Inalámbrico e Interfaz gráfica (software)

Es importante mencionar que el mouse que se usa es inalámbrico para evitar que la persona se enrede con los cables si se tuviera que conectar a la PC. Una sugerencia para el mouse es que sea óptico, ya que responde más fácil a los movimientos que realiza la persona.

El mouse responde a movimientos muy ligeros a la derecha, a la izquierda y de un clic. Con estas tres acciones, la persona escoge la actividad que quiere. Se desarrolló un programa en Visual Basic, presentando una interfaz gráfica donde claramente le muestra al usuario un menú con las opciones que puede realizar.

Debido a que muchas personas con paraplejia/triplejia tienen movimientos de rebote involuntarios al mover la mano, en el programa se está eliminando este problema al detectar solamente el primer movimiento que se llevo a cabo. Si la persona se equivoca, con tan solo un clic en el mouse puede regresar al menú evitando que la acción indeseada se active (en los primeros 5 segundos después del movimiento).

Para la parte del SMS, se envía un mensaje de texto a un celular por medio de la página de Internet de la compañía UNEFON.

Módulo 2 Hardware
En este módulo estamos contemplando todos los circuitos que nos sirven para que las acciones se realicen, así como el foco, la alarma y la televisión.

Hardware

Etapa de la alarma sonora

La frecuencia audible para el ser humano se encuentra entre 20Hz y 20KHz, por lo que nosotros tomamos un valor dentro de este rango, 500Hz. Se debe aplicar una señal cuadrada a la frecuencia antes mencionada, para que la bocina (buzzer) trabaje bien. Esta señal la estamos generando con el microcontrolador durante un segundo.

Debido a que la corriente que entrega el microcontrolador (aproximadamente 1mA) no es suficiente para la bocina (1A), se uso un amplificador de corriente con un arreglo Darlington de transistores (figura 3).

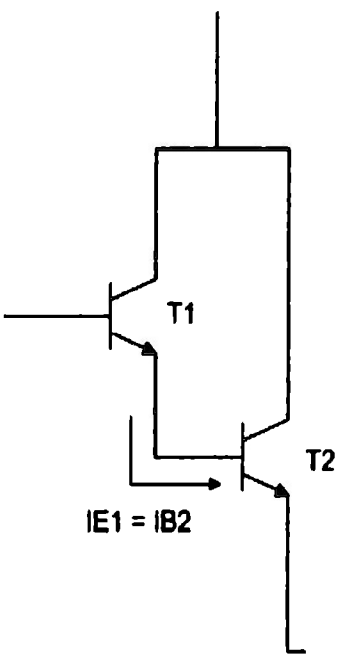


Figura 3 Arreglo Darlington

Explicación del arreglo Darlington

En este arreglo podemos ver que el transistor T1 entrega la corriente que sale por su emisor a la base del transistor T2.

La ecuación de ganancia de un transistor típico es:

$$E = \beta \times IB \qquad (2.0)$$

Donde: IE = corriente de emisor
β = ganancia
IB = corriente de base

Entonces:

Ecuación del primer transistor:

$$IE1 = \beta 1 \times IB1 \qquad (2.1)$$

Ecuación del segundo transistor:

$$IE2 = \beta 2 \times IB2 \qquad (2.2)$$

Observando la figura 3, la corriente de emisor del transistor T1 es la misma que la corriente de base del transistor T2.

Entonces:

$$IE1 = IB2 \qquad (2.3)$$

Utilizando la ecuación (2.2) y la ecuación (2.3)

$$IE2 = \beta 2 \times IB2 = \beta 2 \times IE1 \qquad (2.4)$$

Reemplazando en la ecuación 2.4 el valor de IE, se obtiene la ecuación final de ganancia del transistor Darlington.

$$IE2 = \beta 2 \times \beta 1 \times IB1 \qquad (2.5)$$

Como se puede deducir, este amplificador tiene una ganancia mucho mayor de corriente que la de un transistor normal, pues aprovecha la ganancia de los dos transistores (las ganancias se multiplican).

Para nuestro caso en específico, la multiplicación de ganancias nos da un valor de 1000. El primer transistor es un 2N222 con una β = 50, el segundo transistor es el TIP31 con una β =20. Ahora bien, si teníamos 1mA de corriente entregada por el micro, después de este arreglo (ganancia de 1000) obtendremos la corriente de 1A para aplicarla a la bocina.

Del circuito en la figura 4 podemos calcular la R:

$$\frac{5v - 1.4v}{1mA} = 3600 \qquad (2.6)$$

El valor comercial más próximo es de 3.3K.

El 1.4v se refiere a la caída de voltaje debido a los transistores.

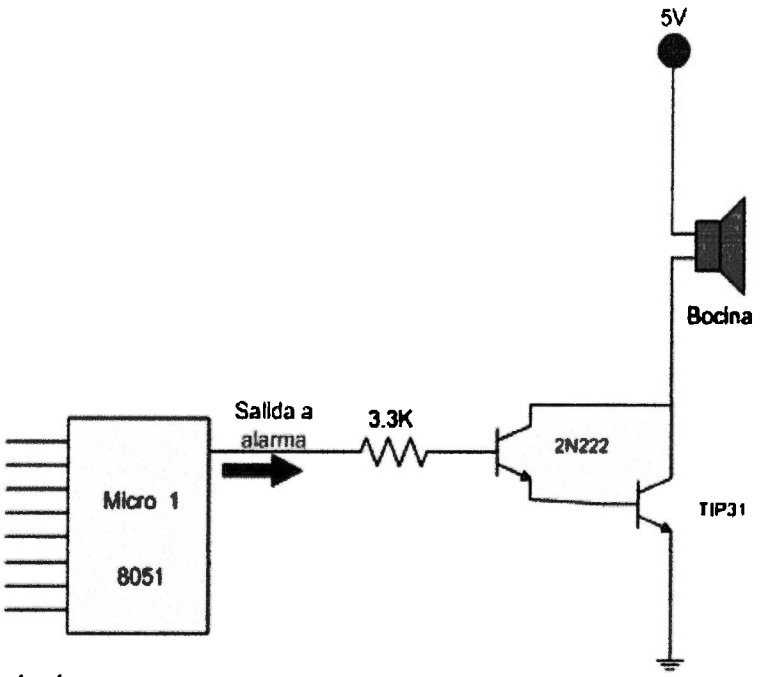


Figura 4 Circuito para activar la alarma sonora

Etapa de la luz

Debido a que se está usando la línea de CA (120V, 60Hz) necesitamos un dispositivo de potencia llamado Triac, y para manejar el Triac un acoplador óptico (MOC3011). Para tener una idea general de cómo funcionan, vamos a explicarlos a más detalle.

Los acopladores ópticos (Figura 5) se diseñan para obtener aislamiento eléctrico completo entre un circuito de entrada y un circuito de salida. El objetivo normal del aislamiento es proporcionar protección para los efectos de las corrientes transitorias de alto voltaje, sobrecargas o ruido de bajo nivel que pudieran dar por resultado una salida errónea o daño del dispositivo.

Los acopladores ópticos también permiten circuitos de acoplamiento con diferentes niveles de voltaje, tierras distintas, etc. Para el circuito integrado usado en el proyecto, el moc3011, el circuito de entrada es un LED y el de salida el fototriac. Cuando el voltaje de entrada polariza en directa al LED, la luz transmitida al fototransistor enciende a éste, produciendo corriente a través de la carga externa. Este dispositivo se diseña para aplicaciones en las que se requiere disparo aislado del triac, como la interrupción de una línea de 120V de ca desde una entrada de bajo nivel.

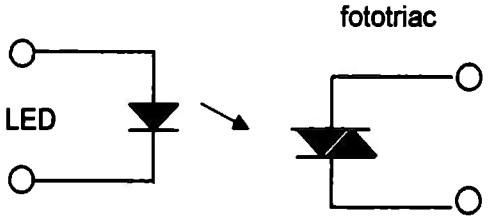


Figura 5 Acoplador óptico

El Triac (Figura 6) es un dispositivo semiconductor que pertenece a la familia de los dispositivos de control por tiristores. El triac es en esencia la conexión de dos tiristores en paralelo pero conectados en sentido opuesto y compartiendo la misma compuerta El triac puede encenderse mediante un pulso de corriente de compuerta y no requiere voltaje de ruptura para iniciar la conducción. Puede concebirse como dos SCR conectados en paralelo y en direcciones opuestas, con una terminal de compuerta común.

El triac es capaz de conducir corriente en cualquier dirección cuando se le dispara a encendido, dependiendo de la polaridad del voltaje a través de sus terminales MT1, MT2. El triac deja de conducir cuando la corriente del ánodo cae por debajo de un valor especificado de la corriente de retención IH. La única forma de apagar el triac es reduciendo la corriente hasta un nivel suficientemente bajo.

El triac sólo se utiliza en corriente alterna y al igual que el transistor, se dispara por la compuerta. Como el triac funciona en corriente alterna, habrá una parte de la onda que será positiva y otra negativa. La parte positiva de la onda (semiciclo positivo) pasará por el triac siempre y cuando haya habido una señal de disparo en la compuerta, de esta manera la corriente circulará de arriba hacia abajo (pasará por el transistor que apunta hacia abajo), de igual manera: La parte negativa de la onda (semiciclo negativo) pasará por el triac siempre y cuando haya habido una señal de disparo en la compuerta, de esta manera la corriente circulará de abajo hacia arriba (pasará por el transistor que apunta hacia arriba). Para ambos semiciclos la señal de disparo se obtiene de la misma patilla (la puerta o compuerta). Lo interesante es, que se puede controlar el momento de disparo de esta patilla y así, controlar el tiempo que cada transistor estará en conducción. (recordar que un transistor solo conduce cuando ha sido disparada (activada) la compuerta y entre sus terminales hay un voltaje positivo de un valor mínimo para cada transistor)

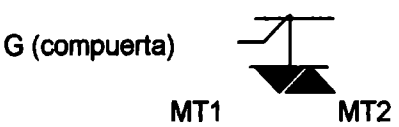


Figura 6 Triac

Dentro de nuestro circuito (Figura 7), se manda un 0 por el microcontrolador cada vez que deseamos prender el foco. Así el transistor usado para aumentar la corriente se pone en corto y deja pasar los 5V para prender el LED. El receptor del MOC3011 excita al Triac mandando un pulso por la compuerta (gate) y es en este momento cuando se enciende el foco.

Cálculo de las Resistencias:

Considerando una corriente de salida del micro 1.8mA (I_{OL}) y un voltaje de 5V.

Para R1

$$R1 = \frac{5V}{1.8mA} = 2.77k \tag{2.7}$$

Con una corriente en la base de .75mA, el voltaje en la R1 es 2.25V

$$V_{R1} = 2.77kX.75mA = 2.25V \tag{2.8}$$

Para R2

$$R2 = \frac{5V - 2.25V}{.75} = 2.66k = 2.7K \tag{2.9}$$

Para R3

$$R3 = \frac{5 - V_{CE} - V_{diodo}}{I_C} = \frac{5v - .1 - 1.2}{10mA} = 370 = 330 \tag{2.10}$$

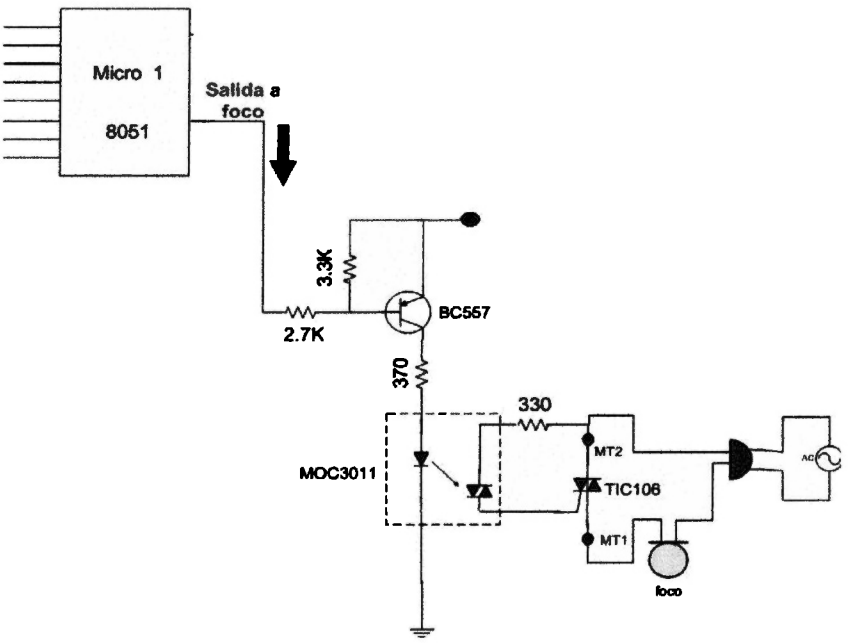
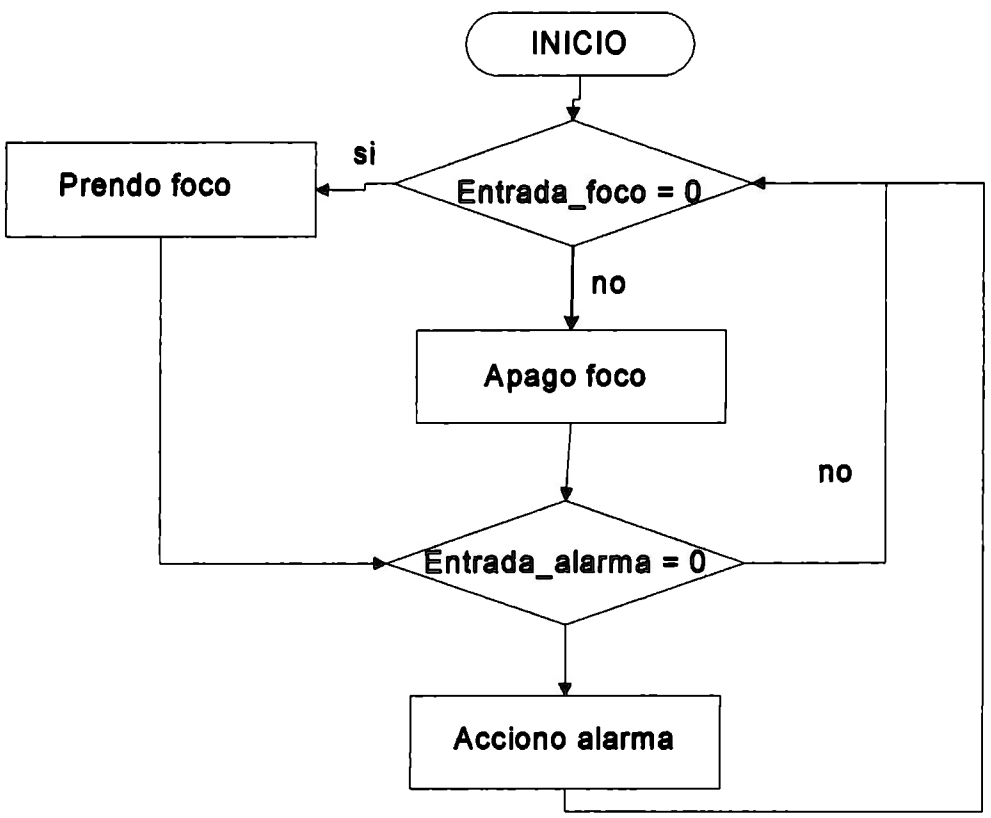


Figura 7 Circuito que sirve para prender/apagar el foco

Algoritmo de la alarma y luz



Nota: El programa completo en ensamblador para la alarma y el foco se encuentra en el anexo de programas.

Etapa de Control Remoto

Los controles remotos utilizan los rayos infrarrojos para mandar las señales a la televisión ya que los infrarrojos no interfieren con otras señales electromagnéticas. La radiación infrarroja es un tipo de radiación electromagnética de mayor longitud de onda que la luz visible, pero menor que la de las microondas. Tiene una frecuencia debajo de la sensibilidad de nuestros ojos, consecuentemente no la podemos ver. Los infrarrojos están asociados al calor debido a que a temperatura normal los objetos emiten espontáneamente radiaciones en el campo de los infrarrojos, incluyendo lámparas, estufas, hornos, la fricción de las manos, incluso el agua caliente y hasta el sol.

Señal enviada por el control remoto
Para evitar que los objetos que emiten rayos infrarrojos interfieran con la señal que se va a mandar a la televisión, tenemos que modular la señal a una frecuencia de 38KHz. Un LED (Diodo Emisor de Luz) infrarrojo emite unos rápidos destellos a una frecuencia de 38 KHz, normalmente (el LED se enciende y apaga 38000 veces por segundo), estos destellos representan un código especial que indica el dispositivo receptor y la acción a realizar. Esa es la señal portadora sobre la que van modulados los pulsos que transmiten información. Así, la señal enviada consiste en pulsos de información, siendo cada uno de ellos una onda cuadrada. El circuito receptor posee un filtro pasa-banda que sólo deja pasar señales a esta frecuencia.

Los principales estándares de modulación que se utilizan en el control remoto son: modulación por ancho de pulso (pulse coded), modulación por duración del espacio (space coded) y modulación por la dirección de transición (RC5)

En la modulación por duración del espacio, la duración del espacio entre los pulsos varía dependiendo si es 1 ó 0 (Figura 8). Los controles remotos de PANASONIC utilizan esta técnica.

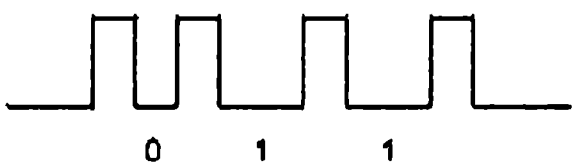


Figura 8 Modulación por duración del espacio

En la modulación RC5 (Figura 9) la dirección de la transición es la que indica si es 1 ó 0 y todos los bits tienen el mismo periodo de tiempo. Philips utiliza esta modulación.

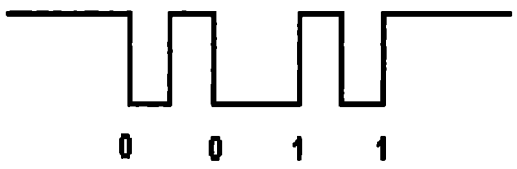


Figura 9 Modulación RC5

El tipo de modulación utilizada en el proyecto fue por ancho de pulso porque nos enfocamos a televisores marca SONY, por la amplia disponibilidad de estos televisores en el mercado a nivel mundial. Cabe mencionar que la programación del código en el microcontrolador puede se adaptada fácilmente si se tiene alguna televisión de otra marca como Philips, Panasonic, etc.

Señal transmitida a los televisores SONY

- Cada paquete está constituido por 12 bits y un header (marca el inicio de la trama)

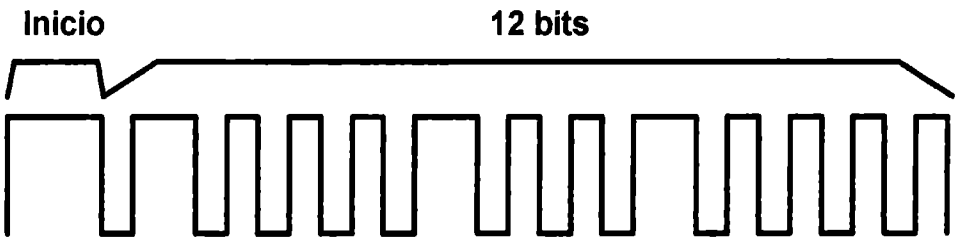


Figura 10 Paquete que contiene el inicio y los 12 bits

- El periodo básico es de T = 600 microsegundos
- La longitud del header es 4T
- El 0 = pulso con longitud de T en bajo, seguido por un espacio de longitud T (señal modulada), por lo tanto tiene una longitud de 1200microsegundos

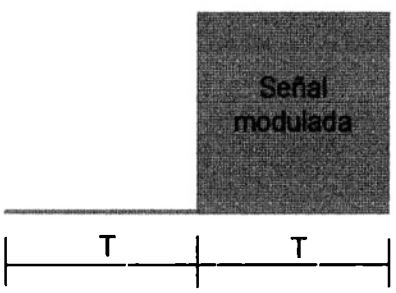


Figura 11 El bit 0

- El 1 = pulso con longitud de T (bajo), seguido por un espacio de longitud 2T (señal modulada), por lo tanto tiene una longitud de 1800 microsegundos.

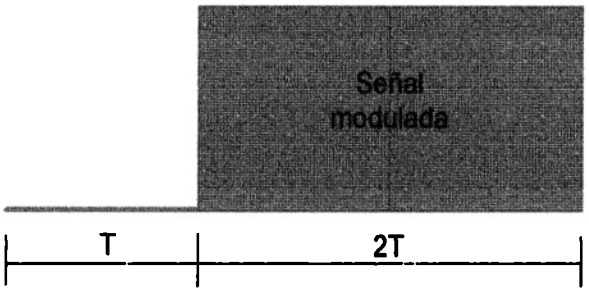


Figura 12 El bit 1

- Los últimos 5 bits representan la dirección y los primeros 7 bits representan el comando.

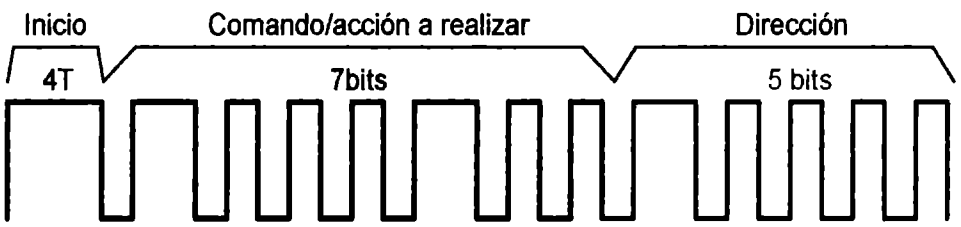


Figura 13 Secuencia completa de la trama para televisores Sony

El código que usa Sony para realizar las diferentes acciones se muestra en la siguiente tabla.

acción	código	acción	código
1	(0) 000 0000	Enter	(11) 000 1011
2	(1) 000 0001	Channel up	(16) 001 0000
3	(2) 000 0010	Channel down	(17) 001 0001
4	(3) 000 0011	Volume up	(18) 001 0010
5	(4) 000 0100	Volume down	(19) 001 0011
6	(5) 000 0101	Power on	(21) 001 0101
7	(6) 000 0110	Power off	(47) 010 1111
8	(7) 000 0111	Mute	(20) 001 0100
9	(8) 000 1000		
0	(9) 000 1001		

Tabla 1 Acciones y códigos de las televisiones SONY

Nosotros solamente nos enfocamos en ON/OFF, CH+, CH-, Vol + y Vol -.

Para generar las señales deseadas y transmitirlas al televisor, usamos el microcontrolador. Por uno de sus puertos (P0), recibimos del puerto paralelo de la PC el código que nos indica que acción desea realizar el usuario. Los códigos son los siguientes:

P0.8	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	
0	0	0	0	0	0	0	0	0	Power
0	0	0	0	0	0	0	0	1	Vol +
0	0	0	0	0	0	0	1	0	Vol -
0	0	0	0	0	0	0	1	1	Ch +
0	0	0	0	0	0	1	0	0	Ch –

Para saber el momento exacto en el que el usuario desea usar la TV, checamos otro bit del puerto del micro, de tal forma que cuando se tuviera 1 en esa entrada, se lee el código y envía la señal al televisor, pero si la entrada era 0, no se realizaba acción alguna.

El algoritmo del control remoto se muestra en la figura 14

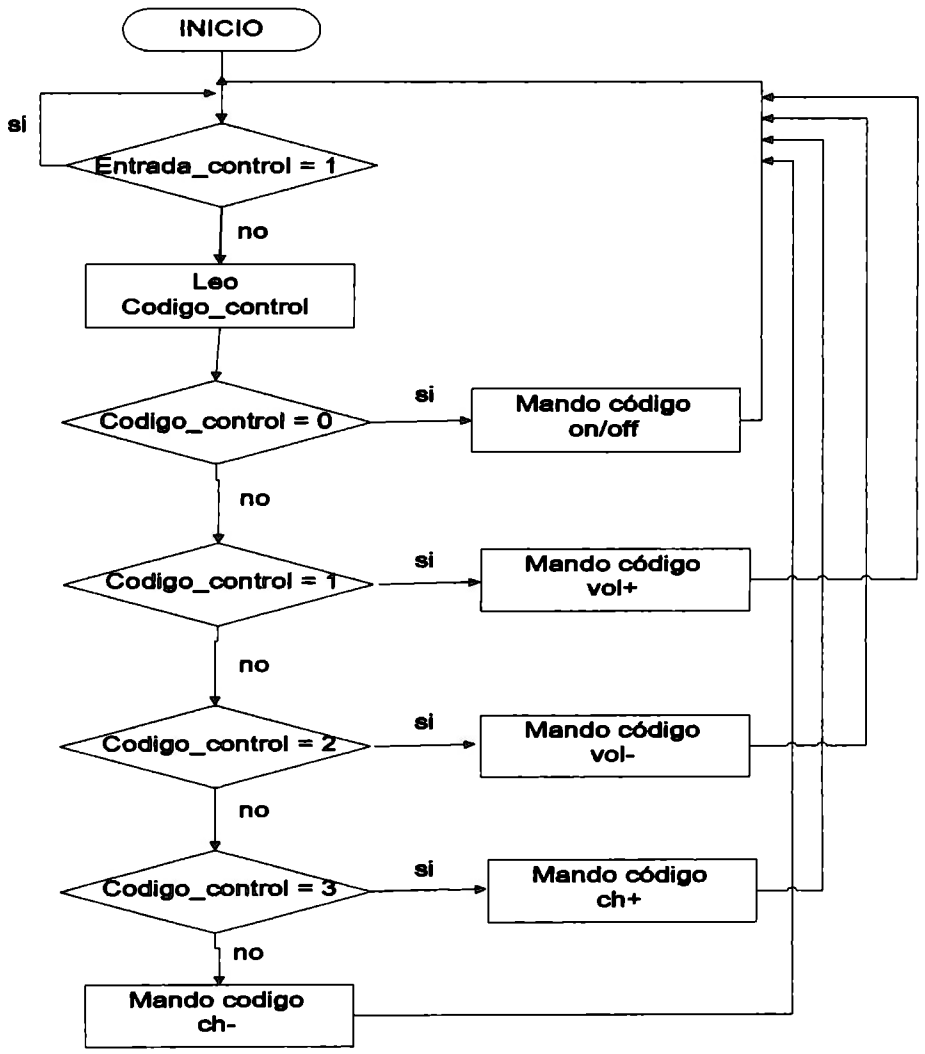


Figura 14 Algoritmo del control remoto

Para verificar que las señales a mandar fueran las correctas utilizamos un osciloscopio, conectado a la salida del micro.

Ch- 1000100 10000

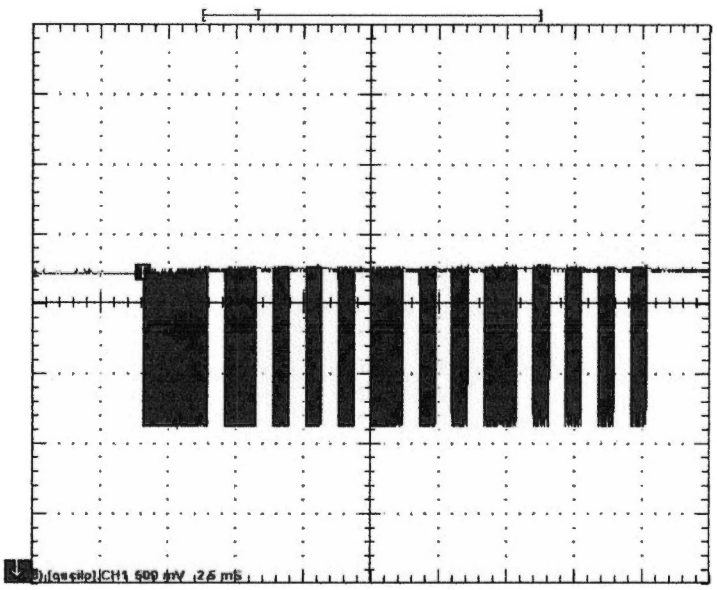


Figura 15 Señal de bajar canal

Ch+ 0000100 10000

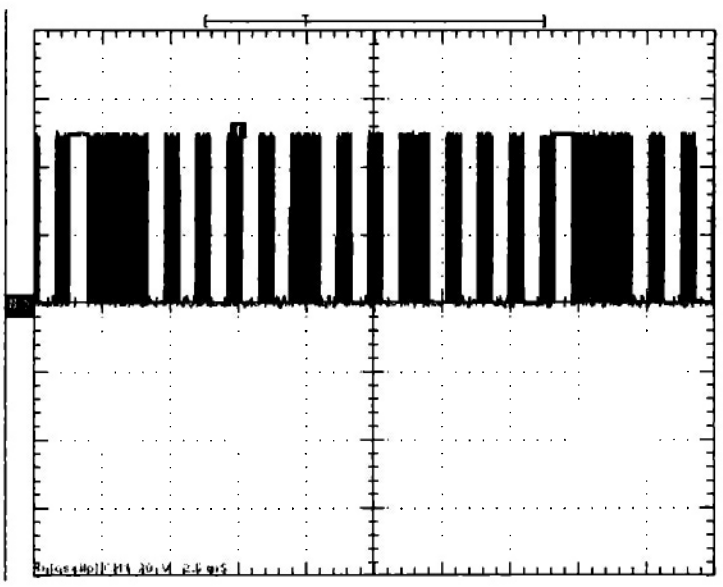


Figura 16 Señal de subir canal

Vol- 1100100 10000

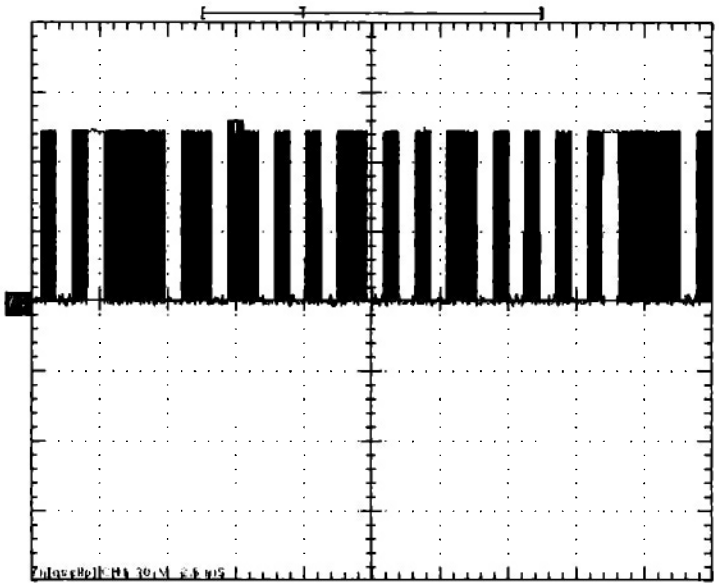


Figura 17 Señal de bajar volumen

Vol + 0100100 10000

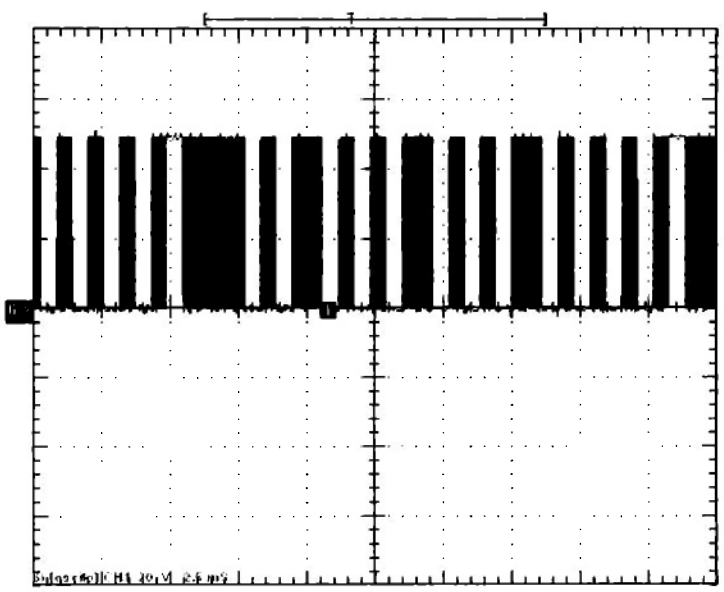


Figura 18 Señal de subir volumen

On/off 1010100 10000

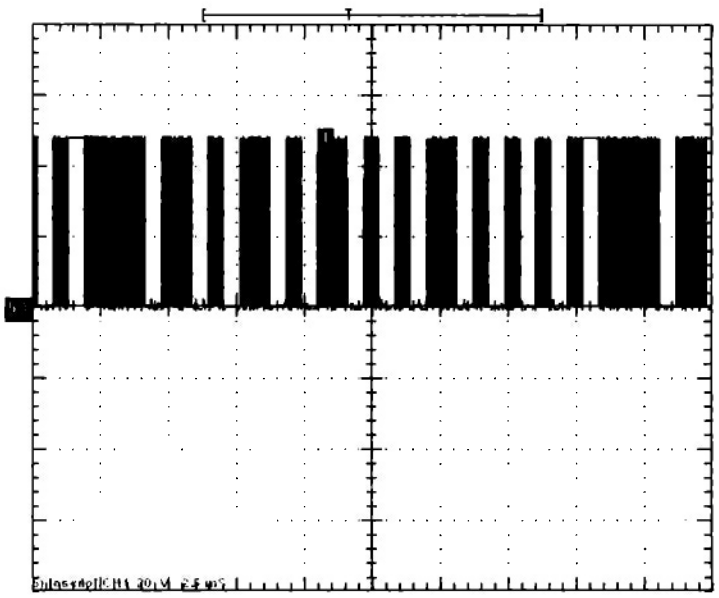


Figura 19 Señal de prender/apagar

La señal que sale del micro llega al LED infrarrojo transmisor

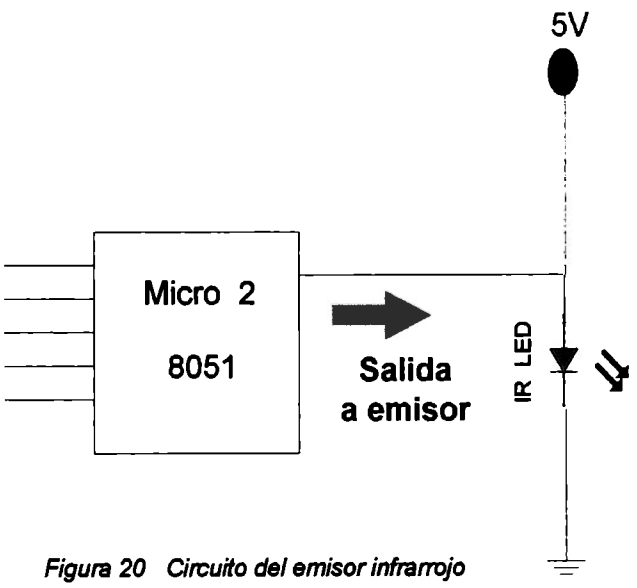


Figura 20 Circuito del emisor infrarrojo

SOFTWARE

SMS

El servicio de mensajes cortos o SMS (*Short Message Service*) es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos (también conocidos como mensajes de texto, o más coloquialmente, textos o incluso *txts* o *msjs*) entre teléfonos móviles, teléfonos fijos y otros dispositivos de mano. SMS fue diseñado originariamente como parte del estándar de telefonía móvil digital GSM, pero en la actualidad está disponible en una amplia variedad de redes, incluyendo las redes 3G.

En esta parte del proyecto implementamos un programa en c++, el cual nos va a permitir mandar un SMS a un teléfono móvil, gracias al portal que UNEFON tiene para mandar SMS vía Internet.

Pero porque elegimos usar C++, en lugar de seguir con Visual Basic, aquí están las siguientes razones:

En primer lugar, cabe destacar la compatibilidad: C++ nos permite usar C (sin objetos) o C++. Yo creo que la compatibilidad ha sido siempre una de las grandes bases de Microsoft y como nosotros nos estamos dirigiendo al sistema operativos Windows desarrollado por Microsoft es por eso que necesitábamos un programa que fuera compatible con este sistema operativo.

La segunda razón mas fuerte para usar C++ son las MFC (las Microsoft Foundation Classes). Estas son clases construidas por Microsoft (aunque su código está disponible y es abierto). Estas nos ahorran mucho trabajo ya que podemos usarlas directamente en lugar de tener que construirnos nuestras propias clases. Normalmente no usaremos la clase tal y como está, sino que derivaremos otra que se adapte mejor a nuestras necesidades. La ATL (Active Template Library) es otra de las bases C++.

Comparando C++ vs Visual Basic podemos decir que un programa bien hecho en C++ falla muy raramente y, si lo hace, puedes encontrar el fallo. Un programa en Visual Basic puede fallar al hacer una cosa que ha hecho treinta veces correctamente. Ese fallo sólo lo puedes resolver reiniciando el ordenador o haciendo algo que tiene una relación remota con Visual Basic: por ejemplo, cerrando programas para liberar memoria o que crean conflictos con Visual Basic.

Así pues, Visual Basic tiene sus fallos; pero hacer un programa en Visual Basic es mucho más sencillo que hacerlo en C++. Esto te permite concebir programas muy complejos que hagan cosas muy complicadas. Así pues, nosotros llegamos a la conclusión de que Visual Basic es un lenguaje para hacer cosas grandes y C++ como un lenguaje para hacer cosas pequeñas donde la eficacia y la seguridad sean importantes.

Para la realización de este programa usamos diferentes librerías y funciones que tiene C++, las cuales son las siguientes:

HWND

En el aparece un variable de tipo HWND, esta variable nos es útil ya que con ella podemos saber en que ventana estamos trabajando. El sistema Windows, trabaja con un

identificador para cada ventana y es única para cada ventana. ActiveX necesitan de un HWND de tu forma, un botón o una caja para operar en la ventana. Pero ahora con HWND activeX nos ayuda a saber que ventana estamos manejando. Uno puede saber quien es el padre, el hijo y el HWND de la próxima ventana.

En resumen activeX, nos ayuda a tener el control de la ventana que vamos a manejar en el programa, por si se abre otra ventana al mismo tiempo, que el programa este al tanto de que ventana es en la que tiene que trabajar.

HWND activeX tiene los siguientes procesos:

GetParent(h Window), este método nos regresa el control de la ventana padre.

GetChild(hWindow), este método nos regresa el primer hijo de la ventana.

GetNext(hWindow), este método nos regresa el hermano de debajo de la ventana por así decirlo.

GetPrevius(Window), este método nos regresa el hermano de arriba.

hWindow, hParentWindow o hChildAfter, son ventanas con las que se tiene un control, si el metodo regresa (Cero) , eso significa que hay un error o que no existe ninguna ventana.

HWnd, El control de la ventana activeX que tenemos, lo podemos pasar en cualquie momento por un método.

KEYBD_EVENT

Esta función nos fue de gran ayuda ya que la base del programa es llenar un cuestionario en Internet de manera automática, es decir sin que el usuario tenga que escribir el número telefónico ni el mensaje que se va a mandar. Esto lo hacemos con la función Keybd_event, esta función sintetiza un teclado. El driver del teclado manda una interrupción a la función keybd_event.

En resumen, lo que nosotros implementamos para el programa fue mandar a llamar esta interrupción (KEYBD_EVENT) vía software y así simular el teclado físico que tenemos en cualquier computadora y poder llenar la forma en Internet.

La sintaxis de esta función es así:

VOID keybd_event(BYTE bvk, BYTE bscan, DWORD dwFlags, PTR dwExtraInfo);

Donde los parámetros son los siguientes:

Parámetros:

bVk Este parámetro es el código de la tecla, el valor tiene que estar dentro del rango 1 al 254. Para ver la lista completa de los códigos ver anexo de KEYBD_event.

bScan Este parámetro usualmente no se usa.

dwFlags Este parámetro puede ser KEYEVENT_EXTENDEDKEY, que es para decirle al programa que el código viene con un valor en especifico de 0xE0, KEYEVENTF_KEYUP, este parámetro no dice que la tecla se libera después de oprimirla y no que se queda oprimida.

dwExtraInfo En este parámetro se recibe información adicional del teclado.Esta función no regresa ningún valor.

El programa lo puedes encontrar en la parte de anexos.

¿PORQUE VISUAL BASIC?

Visual-Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en una gran parte a partir del diseño de una interface gráfica. En una aplicación Visual - Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interface gráfica.

Es por tanto un termino medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias. Ya que no podemos decir que VB pertenezca por completo a uno de esos dos tipos de programación, debemos inventar una palabra que la defina : PROGRAMACION VISUAL.

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

- Creación de una interfase de usuario. Esta interfase será la principal vía de comunicación hombre máquina, tanto para salida de datos como para entrada. Será necesario partir de una ventana - Formulario - a la que le iremos añadiendo los controles necesarios.

- Definición de las propiedades de los controles - Objetos - que hayamos colocado en ese formulario. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para qué sirven.

- Generación del código asociado a los eventos que ocurran a estos objetos. A la respuesta a estos eventos (click, doble click, una tecla pulsada, etc.) le llamamos Procedimiento, y deberá generarse de acuerdo a las necesidades del programa.

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

En nuestro programa utilizamos llamadas a sistema y uso de API´s para llamadas al sistema, uso de hardware, etc. Las cuales se explicaran a continuación.

¿Que es un DLL?

Dll - Dynamic Link Library ("Biblioteca de vínculos dinámicos" es un archivo que contiene funciones que se pueden llamar desde aplicaciones u otras Dll. Los desarrolladores utilizan las Dll para poder reciclar el código y aislar las diferentes tareas. Las Dll no pueden ejecutarse directamente, es necesario llamarlas desde un código externo.

Visual Basic, es un entorno de programación en ambiente gráfico, pese a su potencia y facilidad no dispone en forma directa de instrucciones que permitan obtener del puerto paralelo.

Esta aparente desventaja se elimina si se utilizan funciones que usen algunas características de Windows, este tipo de problemática y su tentativa solución, se desarrolla una DLL capaz de resolver el acceso al puerto paralelo, a la DLL se le incorporan rutinas, tanto para obtener datos como para introducirlos, del mundo exterior hacia la PC.

API

Las API ("application programming interface" o "interfaz de programación de la aplicación") de Windows son un conjunto de funciones propias del sistema operativo que puedes usar en tus programas, como por ejemplo, la función "SetWindowPos" que te permite mantener una ventana por encima de otras.

Todas estas funciones de Windows están escondidas dentro de unas cuantas DLLs ("Dynamic Link Library" o "Librería de Enlace Dinámico") agrupadas según la función que realizan:

Kernel32.dll – operaciones de archivos y gestión de memoria.

Gdi32.dll – operaciones gráficas

User32.dll – maneja la parte de la interacción con el usuario.

En otras palabras una API es un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas *inferiores* y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

API´s utilizadas

HWND (IDENTIFICADOR DE VENTANAS).

El hwnd es un numero, ahora ¿que función cumple ese numero?, imaginemos que tenemos 3 aplicaciones abiertas Word, Excel, y PowerPoint, y la tenemos abierta al mismo tiempo, cuando damos clic a una de ellas en la pantalla, Windows sabe a que programa se da clic gracias al HWND, cada ventana (ojo ventana no programa, ya explicare esto con mas detalle), posee un numero único al momento de ejecutarse, este numero único entra en un registro de ventanas abiertas en el momento de ejecutarlo, cuando se le da clic a una ventana o cualquier otro mensaje él (Windows) sabe para quien va el mensaje ya que el tiene un registro de las ventanas abiertas, y mediante un algoritmo ejecuta la función encargada de controlar el mensaje.

Mouse_event. :User32

mouse_event

Api: Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, ByVal dx As Long, ByVal dy As Long, ByVal cButtons As Long, ByVal dwExtraInfo As Long)

Esta función es la encargada de simular algun evento del mouse, como primer parametro "dwFlags" especificamos que acción vamos a realizar, para ello dicho parametro puede tomar estas constantes:

Public Const MOUSEEVENTF_LEFTDOWN = &H2 : Con este simulamos que el botón derecho del mouse fue presionado.

Public Const MOUSEEVENTF_LEFTUP = &H4 : Con este simulamos que el botón derecho del mouse fue dejado de presionar.

Public Const MOUSEEVENTF_MIDDLEDOWN = &H20 : Con este simulamos que el botón medio del mouse fue presionado.

Public Const MOUSEEVENTF_MIDDLEUP = &H40 : Con este simulamos que el botón medio del mouse fue dejado de presionar.

Public Const MOUSEEVENTF_MOVE = &H1 : Con esta constante indicamos que movemos el mouse.

Public Const MOUSEEVENTF_ABSOLUTE = &H8000 : Especificamos "el como" se realiza el movimiento del mouse.

Public Const MOUSEEVENTF_RIGHTDOWN = &H8 : Con este simulamos que el botón derecho del mouse fue presionado.

Public Const MOUSEEVENTF_RIGHTUP = &H10 : Con este simulamos que el botón derecho del mouse fue dejado de presionar.

Public Const MOUSEEVENTF_WHEEL = &H800 : Solo usado en NT, especifica que la rueda que poseen algunos mouse, ha sido movida, la cantidad del movimiento especificado en el parámetro "cButtons".

Como segundo y tercer parámetro especificamos las coordenadas del puntero del mouse, en donde se realizara la acción especificada arriba.

El el cuarto parámetro especificamos la cantidad de movimiento de la rueda del mouse, si fue especificado en "dwFlags" la constante MOUSEEVENTF_WHEEL. Si el valor es positivo implica que la rueda fue movida hacia delante, si es negativo la rueda fue movida hacia atrás.

Y como último parámetro pasamos una información extra, asociada al evento del mouse.

Vamos a explicar la constante MOUSEEVENTF_ABSOLUTE, ya que en su explicación puse : ("el como" se realiza el movimiento del mouse"). Imaginemos que el puntero se encuentra en la coordenada (pixeles) 400x300, y llama la función mouse_event, con dx = 100 y dy = 50, si no especifico dicha constante, la nueva coordenada del mouse es 500x350, es decir, se desplaza, en cambio si uso dicha constante en la función, la nueva coordenada sería 100x50

En nuestro ejemplo usamos la constante, esa es la razón del por que el mouse cada 500 milisegundos cambia de posición:

```
mouse_event MOUSEEVENTF_ABSOLUTE Or MOUSEEVENTF_MOVE, Int(Rnd() * Screen.Width * 15), Int(Rnd() * Screen.Height * 15), 0&, 0&
```

Ustedes diran ¿Para que simular movimientos, clic, etc, del mouse, o simular las funciones del teclado?. En nuestro caso porque necesitamos que se hagan movimientos del mouse para que la persona parapléjica no necesite mayores movimientos en el uso de la interfaz.

GetWindowRect: User32

```
Api: Declare Function GetWindowRect Lib "user32" (ByVal hwnd As Long, lpRect As RECT) As Long
```

Esta función lo que hace es tomar las coordenadas de "pantalla" de una ventana. Como primer parámetro le pasamos el "hwnd" de la ventana a la cual queremos saber su ubicación, y como segundo parámetro le pasamos una variable del tipo RECT en donde se almacenaran dichas coordenadas.

En nuestro proyecto la usamos para conocer las coordenadas de pantalla del botón:

```
GetWindowRect Command1.hwnd, r

r.Left = r.Left + (X \ 15)

r.Top = r.Top + (Y \ 15)
```

Es decir, luego a esas coordenadas le sumamos, el lugar donde se presiono el botón secundario del mouse, divido entre 15 ya que recuerden que estas coordenadas están en Twips, y 1 Twip equivale a 15 píxeles, es esa la razón del porque el menú se abre justo en el puntero del mouse.

GetSystemMetrics:user32

Api: Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long

Con esta función podemos retornar información métrica (es decir, ancho y alto de distinto componentes) y además algunas configuraciones del sistema. Todas las medidas se encuentran especificadas en píxeles.

La información que queremos buscar, se le pasa como constante en el parámetro "nIndex" y lo que retorna la función es el valor que estábamos buscando. ".

```
Private Const SM_ARRANGE = 56 ' Flag que sirve para conocer como el sistema ordena las ventanas minimizadas.

Private Const SM_CLEANBOOT = 67 ' Con este parámetro podemos saber como se inicio el sistema. Si la función retorna 0 = Booteo normal, 1 = Arranque en modo seguro, 2 = Arranque modo seguro con compatibilidad de red.

Private Const SM_CMONITORS = 80 ' Sirve para conocer la cantidad de monitores, conectados al sistema.

Private Const SM_CMOUSEBUTTONS = 43 ' Sirve para conocer la cantidad de botones presente en el mouse.

Private Const SM_CXBORDER = 5

Private Const SM_CYBORDER = 6 ' Ancho y alto del borde de las ventanas.

Private Const SM_CXCURSOR = 13

Private Const SM_CYCURSOR = 14 ' Ancho y alto del cursor de windows

Private Const SM_CXDLGFRAME = 7

Private Const SM_CYDLGFRAME = 8 ' Hace lo mismo que SM_CXFIXEDFRAME y SM_CYFIXEDFRAME

Private Const SM_CXDOUBLECLK = 36

Private Const SM_CYDOUBLECLK = 37 ' Ancho y alto del rectangulo localizado alrededor del primer click.

Private Const SM_CXEDGE = 45
```

Private Const SM_CYEDGE = 46 ' Ancho y alto del borde 3-D

Private Const SM_CXFIXEDFRAME = SM_CXDLGFRAME

Private Const SM_CYFIXEDFRAME = SM_CYDLGFRAME ' Ancho y alto del rectangulo o "frame" alrededor del perimetro de la ventana que presenta "Caption" pero no es redimensionable.

Private Const SM_CXFULLSCREEN = 16

Private Const SM_CYFULLSCREEN = 17 ' Ancho y alto del area cliente, establecido por una ventana maximizada.

Private Const SM_CXHSCROLL = 21

Private Const SM_CYHSCROLL = 3 ' Ancho de la flecha presente en un scroll horizontal, y alto del scroll horizontal.

Private Const SM_CXHTHUMB = 10 ' Ancho del "thumb" del scroll horizontal. Thumb es la barra por donde se desplaza el cuadrito que vemos en una barra de desplazamiento.

Private Const SM_CYVTHUMB = 9 ' Alto del "thumb" del scroll vertical.

Private Const SM_CXICON = 11

Private Const SM_CYICON = 12 ' Ancho y alto por defecto de un icono.

Private Const SM_CXMAXIMIZED = 61

Private Const SM_CYMAXIMIZED = 62 ' Ancho y alto por defecto de una ventana cuando se maximiza.

Private Const SM_CXMENUCHECK = 71

Private Const SM_CYMENUCHECK = 72 ' Ancho y alto por defecto de la casilla de verificacion colocada en el menu.

Private Const SM_CXMIN = 28

Private Const SM_CYMIN = 29 ' Ancho y alto mínimo que puede tomar una ventana.

Private Const SM_CXMINIMIZED = 57

Private Const SM_CYMINIMIZED = 58 ' Ancho y alto de una ventana minimizada.

Private Const SM_CXSCREEN = 0

Private Const SM_CYSCREEN = 1 ' Ancho y alto de la dimensión de la pantalla

Private Const SM_CXSIZE = 30

Private Const SM_CYSIZE = 31 ' Ancho y alto de los botones presenta en el "Caption" o barra de título.

Private Const SM_CXSMICON = 49

Private Const SM_CYSMICON = 50 ' Ancho y alto recomendados para los iconos pequeños

Private Const SM_CXVSCROLL = 2

Private Const SM_CYVSCROLL = 20 ' Alto de la flecha presente en un scroll vertical, y ancho del scroll vertical.

Private Const SM_CYCAPTION = 4 ' Alto de la barra de título o "Caption" de una ventana

Private Const SM_CYMENU = 15 ' Alto de una simple línea del menu bar.

Private Const SM_MOUSEPRESENT = 19 ' Indica mediante TRUE si hay un mouse instalado.

Private Const SM_MOUSEWHEELPRESENT = 75 ' Indica mediante TRUE o FALSE la existencia de la ruedita que poseen algunos mouse.

Estas son algunas constantes, de todas maneras revisen la documentacion oficial, ya que a medida que salen nuevas versiones de windows dicho listado de constantes puede ir aumentando.

Para la constante SM_ARRANGE, la función puede retornar algunos de los siguientes valores:

Private Const ARW_BOTTOMLEFT = &H0& ' Empieza en la esquina inferior izquierda de la pantalla (posicion por defecto)

Private Const ARW_BOTTOMRIGHT = &H1& 'Empieza en la esquina inferior derecha de la pantalla.

Private Const ARW_HIDE = &H8& ' Esconde la ventana minimizada.

Private Const ARW_TOPLEFT = &H2& ' Empieza en la esquina superior izquierda de la pantalla.

Private Const ARW_TOPRIGHT = &H3& ' Empieza en la esquina superior derecha de la pantalla.

Una vez explicadas las API´s que se utilizaron veremos como las aplicamos a nuestro proyecto.

Utilizamos una clase llamada CmouseEvent la cual hace lo siguiente:

```
Option Explicit

Private Declare Function GetWindowRect Lib "user32" (ByVal hWnd As Long, lpRect As
RECT) As Long
Private Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As
Long
Private Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, ByVal dX As
Long, ByVal dY As Long, ByVal dwData As Long, ByVal dwExtraInfo As Long)
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

' Definición de la estructura del API para un rectángulo.
Private Type RECT
  Left As Long
  Top As Long
  Right As Long
  Bottom As Long
End Type

' Flags used with mouse_event
Private Const MOUSEEVENTF_ABSOLUTE = &H8000& ' absolute move
Private Const MOUSEEVENTF_LEFTDOWN = &H2     ' left button down
Private Const MOUSEEVENTF_LEFTUP = &H4       ' left button up
Private Const MOUSEEVENTF_MIDDLEDOWN = &H20  ' middle button down
Private Const MOUSEEVENTF_MIDDLEUP = &H40    ' middle button up
Private Const MOUSEEVENTF_MOVE = &H1         ' mouse move
Private Const MOUSEEVENTF_RIGHTDOWN = &H8    ' right button down
Private Const MOUSEEVENTF_RIGHTUP = &H10     ' right button up
Private Const MOUSEEVENTF_WHEEL = &H800      ' wheel button rolled

' GetSystemMetrics() codigos
Private Const SM_CXSCREEN = 0
Private Const SM_CYSCREEN = 1

' Para el Uso del scrolol
Private Const WHEEL_DELTA As Long = 120

' variables de ayuda
Private m_ScreenWidth As Long
Private m_ScreenHeight As Long
```

```
Private m_ClickDelay As Long

' Escala Virual para la pantalla
Private Const m_Scale As Long = &HFFFF&

' Direccion del scroll
Public Enum WheelDirections
  meWheelForward = WHEEL_DELTA
  meWheelBackward = -WHEEL_DELTA
End Enum

' ************************************************************
' Inicializar
' ************************************************************
Private Sub Class_Initialize()
  'Guarda las dimensiones de la pantalla en pixeles
  m_ScreenWidth = GetSystemMetrics(SM_CXSCREEN)
  m_ScreenHeight = GetSystemMetrics(SM_CYSCREEN)
  ' Duración default de mouse_down
  m_ClickDelay = 250 'milisegundos
End Sub

' ************************************************************
' Propiedades públicas
' ************************************************************
Public Property Let ClickDelay(ByVal NewVal As Long)
  If NewVal >= 0 Then m_ClickDelay = NewVal
End Property

Public Property Get ClickDelay() As Long
  ClickDelay = m_ClickDelay
End Property

' ************************************************************
' Metodos Públicos
' ************************************************************
Public Sub ButtonPress(ByVal Button As MouseButtonConstants)
  ' Presiona el botoón del mouse en un lugar en específico
  Select Case Button
    Case vbLeftButton, vbMiddleButton, vbRightButton
      Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
    Case vbMiddleButton
      Call mouse_event(MOUSEEVENTF_MIDDLEDOWN, 0, 0, 0, 0)
    Case vbRightButton
      Call mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0, 0)
  End Select
```

```
End Sub

Public Sub ButtonRelease(ByVal Button As MouseButtonConstants)
  ' HAce un release al mouse en un lugar en específico
  Select Case Button
    Case vbLeftButton, vbMiddleButton, vbRightButton
      Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
    Case vbMiddleButton
      Call mouse_event(MOUSEEVENTF_MIDDLEUP, 0, 0, 0, 0)
    Case vbRightButton
      Call mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0)
  End Select
End Sub

Public Sub Click()
  ' Hace un click simulando el tiempo que tarda un humano.
  Call mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0)
  If m_ClickDelay Then
    DoEvents
    Call Sleep(m_ClickDelay)
  End If
  Call mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0)
End Sub

' X/Y se necesita pasar a pixeles
Public Sub ClickAbsolute(ByVal X As Long, ByVal Y As Long)
  ' Mueve el cursor a un destino.
  Call Me.MoveTo(X, Y)
  ' HAce click
  Call Me.Click
End Sub

Public Sub ClickWindow(ByVal hWnd As Long)
  ' Mueve el cursor a una ventana
  Call Me.MoveToWindow(hWnd)
  'Hace click
  Call Me.Click
End Sub

' X/Y se necesita pasar a pixeles
Public Sub MoveTo(ByVal X As Long, ByVal Y As Long, Optional ByVal Absolute As
Boolean = True)
  Dim meFlags As Long

  If Absolute Then
    'Mapea en las mismas corrdenadas usadas por mouse_event.
```

```
    X = (X / m_ScreenWidth) * m_Scale
    Y = (Y / m_ScreenHeight) * m_Scale
    ' EStablece las banderas
    meFlags = MOUSEEVENTF_ABSOLUTE Or MOUSEEVENTF_MOVE
  Else
    'Establece las banderas para movimientos relativos
    meFlags = MOUSEEVENTF_MOVE
  End If

  ' Mueve el cursor a un destino
  Call mouse_event(meFlags, X, Y, 0, 0)
End Sub

Public Sub MoveToWindow(ByVal hWnd As Long)
  Dim X As Long, Y As Long
  Dim r As RECT

  ' Pone el origen en el centro del control.
  Call GetWindowRect(hWnd, r)
  X = r.Left + (r.Right - r.Left) \ 2
  Y = r.Top + (r.Bottom - r.Top) \ 2
  Call Me.MoveTo(X, Y)
End Sub


Public Sub TurnWheel(Optional ByVal Notches As Long = 1, Optional ByVal Direction
As WheelDirections = meWheelBackward)
  Dim dwData As Long

  ' Valida la direccion
  If Direction <> meWheelBackward And Direction <> meWheelForward Then
    Direction = meWheelBackward
  End If

  ' HAce el scrooll
  dwData = Notches * Direction
  Call mouse_event(MOUSEEVENTF_WHEEL, 0, 0, dwData, 0)
End Sub
```

Utilizamos otro modulo llamado IO, el cual solamente hace llamadas al IO.DLL el cual acceso a los puertos de la PC y lo utilizamos de manera global en nuestro programa.

```
Public Declare Sub PortOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As Byte)
Public Declare Sub PortWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As
Integer)
Public Declare Sub PortDWordOut Lib "IO.DLL" (ByVal Port As Integer, ByVal Data As
Long)
Public Declare Function PortIn Lib "IO.DLL" (ByVal Port As Integer) As Byte
Public Declare Function PortDWordIn Lib "IO.DLL" (ByVal Port As Integer) As Long
Public Declare Sub SetPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Public Declare Sub ClrPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Public Declare Sub NotPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As Byte)
Public Declare Function GetPortBit Lib "IO.DLL" (ByVal Port As Integer, ByVal Bit As
Byte) As Boolean
Public Declare Function RightPortShift Lib "IO.DLL" (ByVal Port As Integer, ByVal Val
As Boolean) As Boolean
Public Declare Function LeftPortShift Lib "IO.DLL" (ByVal Port As Integer, ByVal Val
As Boolean) As Boolean
Public Declare Function IsDriverInstalled Lib "IO.DLL" () As Boolean


'*************************************************************
'*********  Function Descriptions  ********************
'*************************************************************

'PortOut -Saca un byte por el puerto establecido.

'PortWordOut - Saca uan palabra de 16 bits por el puerto.

'PortDWordOut - Saca una palabra de 32 bits por el puerto.

'PortIn - Lee un byte por el puerto.

'PortWordIn - Reads a word (16-bits) from the specified port.

'PortDWordIn - lee una palabra de 16 bits por el puerto.

'SetPortBit - Pone en 1 un bit.

'ClrPortBit - Pone en cero un bit.

'NotPortBit - Invierte el bit del puerto.

'GetPortBit - Regresa el estado de un bit en específico.

'RightPortShift - Hace un corriemiento a la derecha.
```

'LeftPortShift - Hace un corrimiento a la izquierda, el MSB lo regresa y el valor pasado es ahora el LSB.
'IsDriverInstalled - Regresa un cero si el io.dll está instalado y funcionando para el sistema operativo.

En nuestro caso utilizamos el puerto paralelo el cual se explicará a continuación:

PUERTO PARALELO

Introducción:

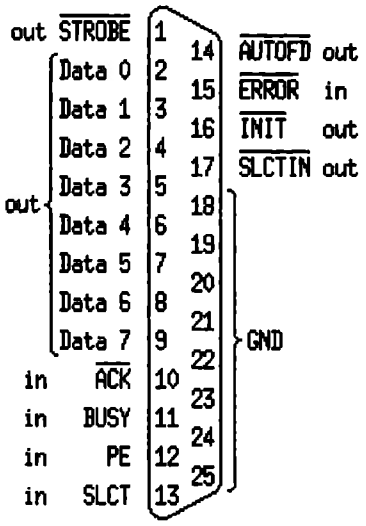
El puerto paralelo de una PC es ideal para ser usado como herramienta de control de motores, relés, LED's, etc. El mismo posee un bus de datos de 8 bits (Pin 2 a 9) y muchas señales de control, algunas de salida y otras de entrada que también pueden ser usadas fácilmente.

Las PC's generalmente poseen solo uno de estos puertos (LPT1) pero hay algunas que se les adicionó el (LPT2).

En reglas generales la dirección hexadecimal del puerto LPT1 es igual a 0x378 (888 en decimal) y 0x278 (632 en decimal) para el LPT2. Esto se puede verificar fácilmente en el setup de la PC.

Breve descripción del puerto paralelo:

El puerto paralelo de un PC posee un conector de salida del tipo DB25 hembra cuyo diagrama y señales utilizadas podemos ver en la siguiente figura:



Si deseamos escribir un dato en el bus de salida de datos (pin 2 a 9) solo debemos escribir el byte correspondiente en la dirección hexadecimal 0X378 (888 en decimal) cuando trabajamos con el LPT1. Los distintos pins (bits) de salida correspondientes al bus de datos no pueden ser escritos en forma independiente, por lo que siempre que se desee modificar uno se deberán escribir los ocho bits nuevamente.

Para leer el estado de los pins de entrada (10, 12, 13 y 15) se debe realizar una lectura a la dirección hexadecimal 0x379 (889 en decimal) si trabajamos con el LPT1 o bien leer la dirección 0x279 (633 en decimal) si trabajamos con el LPT2. La lectura será devuelta en un byte en donde el bit 6 corresponde al pin 10, el bit 5 corresponde al pin 12, el bit 4 corresponde al pin 13 y el bit 3 corresponde al pin 15.

En la siguiente tabla se puede ver lo antedicho en una forma más gráfica:

Escritura: Salida de Datos								
Escritura en dirección 0x378 (LPT1) o 0x278 (LPT2)								
DATO	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DB25	Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin2
CN5	TTL 7	TTL 6	TTL 5	TTL 4	TTL 3	TTL 2	TTL 1	TTL 0
CN4	No usar	HP 6	HP 5	HP 4	HP 3	HP 2	HP 1	HP 0

Lectura: Entrada de Datos								
Lectura en dirección 0x379 (LPT1) o 0x279 (LPT2)								
DATO	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DB 25	No usar	Pin 10	Pin 12	Pin 13	Pin 15	No usar	No usar	No usar
CN6	No usar	Input 3	Input 2	Input 1	Input 0	No usar	No usar	No usar

En el programa principal se utilizaron los módulos descritos anteriormente de la siguiente forma:

```
Option Explicit
Public Out_TTL As Byte        'contiene el BYTE a sacar por el puerto
Public In_Port As Integer     'continen la direccion de lectura del LPT
```

```
Public Out_Port As Integer
Dim c1 As Byte
Dim bit1 As Byte
   Dim bit2 As Byte
   Dim bit3 As Byte
   Dim bit4 As Byte
   Dim bit5 As Byte
   Dim contador2 As Long
   Dim c2 As Long
   Dim c3 As Long
' Win32 API declaraciones
Private Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" (ByVal
lpszName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
Private Declare Function GetWindowRect Lib "user32" (ByVal hWnd As Long, lpRect As
RECT) As Long
Private Declare Sub keybd_event Lib "user32" (ByVal bVk As Byte, ByVal bScan As
Byte, ByVal dwFlags As Long, ByVal dwExtraInfo As Long)
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Private Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, ByVal dX As
Long, ByVal dY As Long, ByVal dwData As Long, ByVal dwExtraInfo As Long)
' API definición de su estructura en forma de rectángulo
Private Type RECT
  Left As Long
  Top As Long
  Right As Long
  Bottom As Long
End Type

' Constantes de sonido
Private Const SND_ASYNC = &H1        ' toca asíncronamente
Private Const SND_ALIAS = &H10000    ' name is a WIN.INI [sounds] entry

' Shell functions
Private Enum SystemKeyShortcuts
  ExplorerNew = &H45  ' Asc("E")
  FindFiles = &H46    ' Asc("F")
  MinimizeAll = &H4D  ' Asc("M")
  RunDialog = &H52    ' Asc("R")
  StartMenu = &H5B    ' Asc("[")
  StandbyMode = &H5E  ' Asc("^") -- Win98 only!
End Enum


' DEfinición de variables
Private m_Mouse As CMouseEvent
Private m_oX As Long
```

```
Private m_oY As Long
Private m_Radius As Long
Private m_Angle As Long
Private m_Offset As Long
Private m_PixIn As Long
Private m_dX As Long
Private m_dY As Long

' indices
Private Const tbRadius = 0
Private Const tbPixels = 1
Private Const tbAngle = 2
Private Const tbOffset = 3
Private Const tbDelay = 4

Private Sub Command1_Click()

Timer1.Enabled = False
Timer1.Interval = 5000

m_Mouse.MoveToWindow command4.hWnd
'
End Sub



Private Sub Command3_Click()
Timer3.Enabled = False
Timer3.Interval = 5000
m_Mouse.MoveToWindow command4.hWnd
End Sub

Private Sub Command3_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
Timer3.Enabled = True

m_Mouse.MoveToWindow Command3.hWnd

End Sub

Private Sub Form_Activate()
 m_Mouse.MoveToWindow command4.hWnd
   Timer1.Enabled = False
   Timer2.Enabled = False
   Timer3.Enabled = False
```

```
End Sub


Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
   Cancel = 1              'evita que se salga mediante la 'X'
End Sub



Private Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)
Dim cont As Integer
cont = 0
Timer1.Enabled = True

m_Mouse.MoveToWindow Command1.hWnd


End Sub

Private Sub Command2_MouseMove(Button As Integer, Shift As Integer, X As Single, Y
As Single)

Timer2.Enabled = True

m_Mouse.MoveToWindow Command2.hWnd


End Sub

Private Sub Command2_Click()
m_Mouse.MoveToWindow command4.hWnd
Timer2.Enabled = False
Timer2.Interval = 5000

End Sub



Private Sub Command4_Click()

 m_Mouse.MoveToWindow Command3.hWnd
```

```
End Sub

Private Sub Form_Initialize()
' SetPortBit Out_Port, 0
 m_Mouse.MoveToWindow command4.hWnd



End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   Label2.Caption = "coordenadas: (" & X & "," & Y & ")"
   If X < 6105 Then
   m_Mouse.MoveToWindow Command1.hWnd



   End If
   If X > 6110 Then

    m_Mouse.MoveToWindow Command2.hWnd
    End If

   End Sub
Private Sub Form_Click()

Print "dio click"
End Sub


Private Sub Form_Load()



  Set m_Mouse = New CMouseEvent


In_Port = &H378
Out_Port = In_Port

 'Port_Reset
```

```
 ' SetPortBit Out_Port, 0
   m_Mouse.MoveToWindow command4.hWnd
   Timer1.Enabled = False
   Timer2.Enabled = False
   Timer3.Enabled = False

End Sub



Private Sub Form_Unload(Cancel As Integer)  'si sale del programa
    Port_Reset                          'entonces reseteo salidas TTL

  Timer1.Enabled = False
  Timer2.Enabled = False
  Timer3.Enabled = False
  Timer1.Interval = 5000
  Timer2.Interval = 5000
  Timer3.Interval = 5000


    End
End Sub



Private Sub Port_Reset()        'procedimiento para resetear salidas TTL
    PortOut Out_Port, 0
End Sub




Private Sub Form_Paint()

End Sub

Private Sub Label10_Click()

End Sub

Private Sub Timer1_Timer()
   Unload Me
```

```
  Load Form3
  Form2.Show


End Sub



Private Sub Timer2_Timer() 'este timer sirve para:
Unload Me
Load Form1
Form1.Show
End Sub


Private Sub Timer3_Timer()
Unload Me
Load Form3
Form3.Show

End Sub
```

CONCLUSIONES

Con el desarrollo de este proyecto deseamos realizar una aplicación funcional y no que sólo se quedara en investigación, y que mejor que enfocarnos a las personas que tienen algún problema motriz. Actualmente en la ingeniería biomédica se estudian diversas formas para desarrollar aparatos que puedan ser utilizados por personas con alguna enfermedad, pero desgraciadamente estos dispositivos llegan a ser muy caros y complicados, por lo que su uso se limita a clínicas u hospitales especializados.

La interfaz para personas parapléjicas, reúne tanto hardware como software que trabajando en conjunto logran dar un poco de independencia a una persona que presenta parálisis en alguna parte de su cuerpo.

Al trabajar en el proyecto profundizamos conocimientos básicos y adquirimos nuevos por medio de la investigación, nos dimos cuenta que no es lo mismo realizar algunos cálculos o estudiar la teoría que llevarlo a la práctica. Siempre hay situaciones fuera de control, y debemos pensar en la mejor solución para resolverlo, basándonos en lo que se ha aprendido.

Entre las principales desventajas de nuestro proyecto es que se necesita una PC para que recupere las acciones que desea realizar la persona discapacitada y posteriormente

mande las señales a un microcontrolador encargado de accionar la luz, la alarma y el control remoto. Pensamos que las mejoras que se pueden realizar es sustituir la pc por una palm, al ser un dispositivo 100% portable. También se podrían añadir más acciones como usar una videograbadora, un estéreo, accionar un ventilador, etc.

BIBLIOGRAFÍA Y REFERENCIAS CONSULTADAS

Boylestad, Robert y Louis Nashelsky. *Electrónica : teoría de circuitos y dispositivos electrónicos*, traducción Carlos Mendoza Barraza, Pearson Educación, México 2003.

Horenstein, Mark N. *Circuitos y dispositivos microelectrónicas*, traducción Gabriel Sánchez García, Prentice-Hall, México, 1997.

Floyd, Thomas. *Dispositivos electrónicos*, tr. Hugo Villagómez Velázquez, Limusa, México, 2003.

www.lesionmedular.org

http://www.aktiva-mx.com/orientacion.htm

http://www.elalmanaque.com/Medicina/lexico/paraplejia.htm

http://salud.discapnet.es/enfermedades+discapacitantes/lesion+de+la+medula+espinal+1
http://www.healthsystem.virginia.edu/UVAHealth/adult_pmr_sp/spcrd.cfm

http://www.neurologia.com.mx/PDFs/REVISTA4-1/

http://atc.ugr.es/~afdiaz/asbm.html

http://www.micropic.arrakis.es/marcos.htm

http://www.bolivar.udo.edu.ve/microinternet/articulos/control%20remoto%20IR%2038KHZ/Control%20Remoto%20IR%2038Khz.htm

http://www.arrl.org/news/features/2004/03/30/1/

http://www.pjrc.com/tech/8051/

http://www.8052.com

http://www.vekoll.vein.hu/~jap/electronic/codec.html

http://roble.cnice.mecd.es/~jsaa0039/

http://www.unicrom.com/

http://users.pandora.be/davshomepage/

ANEXOS

Microcontrolador 8051 de Intel

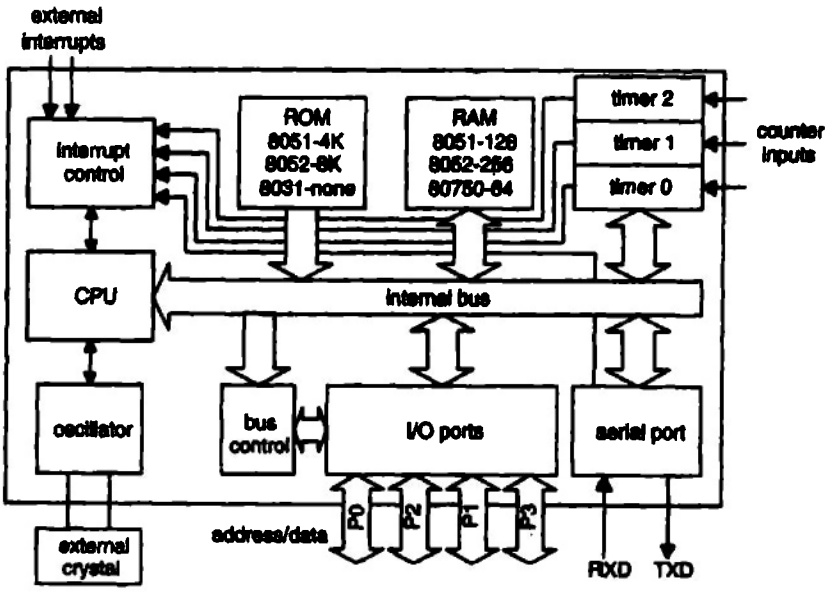
Con el fin de permitir la construcción de circuitos lógicos usando el concepto de lógica programable, los fabricantes de circuitos integrados han visto la necesidad de producir dispositivos, de alta velocidad, con los que se puedan desarrollar funciones lógicas de toda clase.

En estos circuitos, el usuario puede programar, en un sólo chip, funciones que, de otra forma, con la circuitería tradicional de compuertas, utilizarían muchos componentes, además del espacio físico de los mismos.

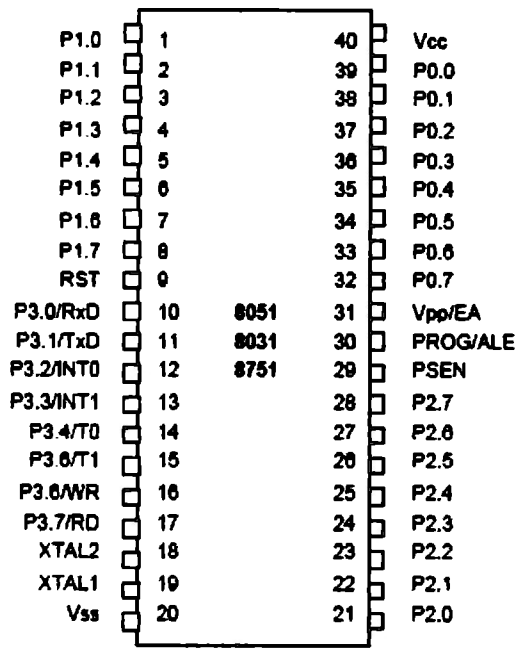
Un microcontrolador es todo un sistema mínimo dentro de un solo dispositivo, lo cual ofrece un enorme panorama hacia el mundo de la compatibilidad. Este dispositivo contiene: CPU (basado principalmente en un microprocesador), puertos paralelos de entrada y salida, puerto serie, timers, contadores, memorias (programa, datos), y en algunos casos hasta convertidores analógicos digitales, todo esto dentro de un solo chip. Básicamente consta de un programa más o menos complejo que da las pautas para realizar un trabajo ayudado por unos sensores y actuadores que recogen la información y transmiten las instrucciones

Características principales del micro 8051

- CPU de 8 bits
- 32 líneas bidireccionales (4 puertos de E/S).
- Memoria de datos (RAM) de 128 bytes.
- Memoria de programa (ROM) de 4KB (8051).
- 4 puertos de E/S con 8 líneas cada uno.
- 1 puerto de E/S serie()UART.
- 2 contadores-temporizadores de 16 bits programables.
- 5 estructuras de interrupción con dos niveles de prioridad
- 1 oscilador para las señales de reloj.
- Capacidad de control de 64 KB de memoria de código.
- Capacidad de control de 64 KB de memoria de datos.
- 2 líneas de interrupción.



Descripción de pines:



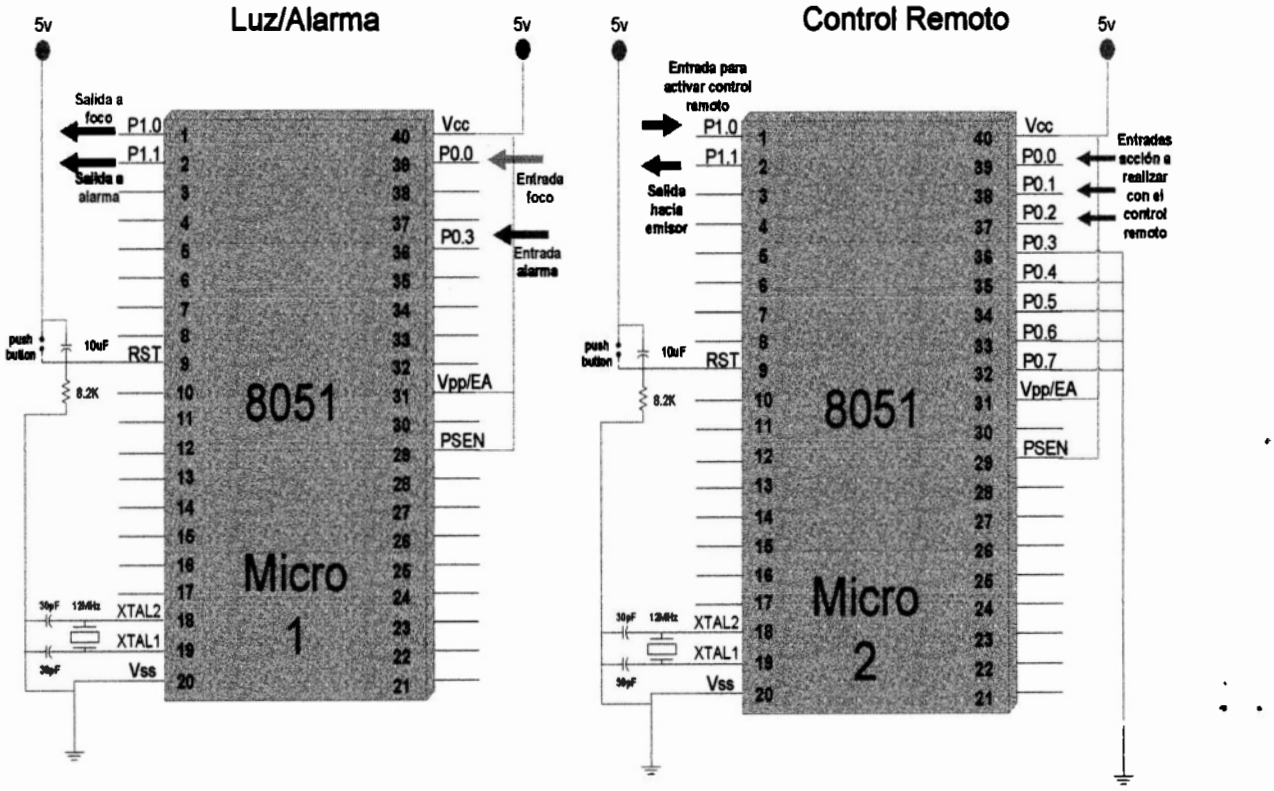
- Vss Es la tierra del microcontrolador y va conectada a 0V
- Puerto 0. Es un puerto bidireccional con salidas en colector abierto. Cuando el puerto tiene 1's escritos, las salidas están flotadas y pueden servir como entradas en alta impedancia. El puerto 0 es también multiplexado para obtener el DATO y la parte baja de la dirección.
- Puerto 1 Es un puerto quasidireccional, cuando se escribe 1's en el puerto, el puerto puede ser utilizado como entrada.
- Puerto 2 Es un puerto quasi-bidireccional con fijadores de nivel internos (pull-up). Cuando se escriben 1's sobre el puerto, las líneas pueden ser utilizadas como entradas o salidas. Como entradas, las líneas que son externamente colocadas en la posición baja proporcionaran una corriente hacia el exterior.
El puerto 2 es utilizado además para direccionar memoria externa. Este puerto, emite el byte más alto de la dirección durante la búsqueda de datos en la memoria del programa externo y durante el acceso a memorias de datos externos que usan direccionamientos de 16 bits. Durante el acceso a una memoria de dato externa, que usa direcciones de 8 bits, el puerto dos emite el contenido del registro del correspondiente a este puerto, que se encuentra en el espacio de funciones especiales.
- Puerto 3 Es un puerto quasi-bidireccional con fijadores de nivel internos (PULL-UP). Cuando se escriben 1's sobre el puerto, las líneas pueden ser utilizadas como entradas o como salidas. Como entradas las líneas que son externamente colocadas en la posición

baja proporcionarán una corriente. El puerto 3 se utiliza además para producir señales de control de dispositivos externos como son los siguientes:

RxD(P3.0): Puerto serie de entrada.
TxD(P3.1): Puerto serie de salida.
INT0(P3.2): Interrupción externa.
INT1(P3.3): Interrupción externa.
T0(P3.4): Entrada externa timer0.
T1(P3.5): Entrada externa timer1.
WR(P3.6): Habilitador de escritura para memoria externa de datos.
RD (P3.7): habilitador de lectura para la memoria externa de datos.

- Reset Una entrada alta en esta línea durante dos ciclos de maquina mientras el oscilador está corriendo detiene el dispositivo. Un resistor interno conectado a Vss permite un alto en la fuente usando solamente un capacitor externo a VCC.
- ALE Address Latch Enable. Un pulso positivo de salida permite fijar el byte bajo de la dirección durante el acceso a una memoria externa. En operación normal, ALE es emitido en un rango constante de 1/6 de la frecuencia del oscilador, y puede ser usada para cronometrar. Note que un pulso de ALE es emitido durante cada acceso a la memoria de datos externos.
- PSEN Program Store Enable. Habilitador de lectura para memoria de programas externos. Cuando el 8031B/8051 está ejecutando un código de una memoria de programas externos, PSEN es activada dos veces cada ciclo de máquina, excepto cuando se accesa la memoria de datos externos que omiten las dos activaciones del PSEN externos. PSEN tampoco es activada
cuando se usa la memoria de programas internos.
- EA External Access Enable. EA debe mantenerse externamente en posición baja para habilitar el mecanismo que elige el código de las localizaciones de la memoria de programas externos, 0000H y 0FFFH. Si EA se mantiene en posición alta, el dispositivo ejecuta los programas que se encuentran en la memoria interna ROM, a menos que el contador del programa contenga una dirección mayor a 0FFFH.
- XTAL1 Crystal 1. Es la entrada del cristal para el circuito oscilador (generador del reloj interno) que amplifica e invierte la entrada.
- XTAL2 18 0 Crystal 2. Es la salida del amplificador oscilador inversor.

Conexión de los micros que se usaron en el proyecto



```
                                        Control Remoto
;*********************************************/
;                                            /
;Programa que controla la TV                 /
;                                            /
;                                            /
;*********************************************/

cont1   EQU     R2
cont2   EQU     R3
cont3   EQU     R4
aux_pto EQU     R7

;entradas

entrada         EQU     P0
uso_control     EQU     P1.0

;bandera para ver cuando se quiere usar el control, con 1 no hay accion, 0 uso

; P8  P7  P6  P5  P4  P3  P2  P1  P0
; 0   0   0   0   0   0   0   0   0    Power
; 0   0   0   0   0   0   0   0   1    Vol +
; 0   0   0   0   0   0   0   1   0    Vol -
; 0   0   0   0   0   0   0   1   1    Ch +
; 0   0   0   0   0   0   1   0   0    Ch -

;salidas

senal   EQU     P1.3


ORG 0

main:
        MOV SP, #70H

esp_usoControl:
        JB uso_control, esp_usoControl

;mandar un pulso de 0 para decir que deseamos usar el control remoto
;si hay un cero, se activo el uso del control remoto

lee_pto:
        mov aux_pto, entrada
        CJNE aux_pto, #0, checa_volmas

;;se desea prender/apagar tv  1010100 10000
        call INICIO_TRAMA
        call UNO
        call CERO
        call UNO
        call CERO
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call CERO
        call CERO
        JMP esp_usoControl

        checa_volmas:
        CJNE aux_pto, #1, checa_volmenos
;;se desea subir el volumen de la tv  0100100 10000

        call INICIO_TRAMA
        call CERO
```

```
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call CERO
        call CERO
        JMP esp_usoControl

        checa_volmenos:
        CJNE aux_pto, #2, checa_chmas
;;se desea bajar el volumen de la  tv  1100100 10000

        call INICIO_TRAMA
        call UNO
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call CERO
        call CERO
        JMP esp_usoControl

        checa_chmas:
        CJNE aux_pto, #3, checa_chmen
;;se desea cambiar el canal + de la  tv  0000100 10000

        call INICIO_TRAMA
        call CERO
        call CERO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call CERO
        call CERO
        JMP esp_usoControl

        checa_chmen:
;;se desea cambiar  el canal - de la  tv  1000100 10000

        call INICIO_TRAMA
        call UNO
        call CERO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call UNO
        call CERO
        call CERO
        call CERO
        call CERO
        JMP esp_usoControl
```

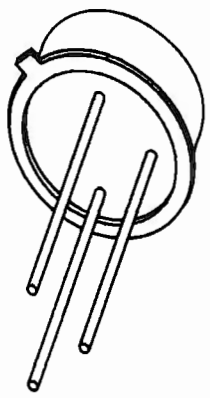
```
; *********    Subrutinas **************

CERO:
        ;601us portadora
        mov cont1, #200
genera:
        CLR senal
        djnz cont1, genera
        ;609us en 1
        mov cont1, #19
genera0:
        call portadora
        djnz cont1, genera0

        RET

UNO:
        ;601us en 1
        mov cont1, #200
        gen:    CLR senal
        djnz cont1, gen

        ;1217us portadora
        mov cont1, #38
genera1:
        call portadora
        djnz cont1, genera1

        RET


;genero la señal cuadrada con un periodo total de 26us
;frecuencia de la portadora es de 40KHZ
;el 0 dura 10us y el 1 dura 16us

INICIO_TRAMA:
        ;601 us en 1
         mov cont1, #200
trama0:
        SETB senal
        djnz cont1, trama0

        ;2404 us
        mov cont1, #4
ciclo1:
        mov cont2, #19
trama1:
        call portadora
        djnz cont2, trama1
        djnz cont1, ciclo1
        RET

portadora:
        ;tengo 27 us

        MOV cont3, #4
por2:
        CLR senal
        DJNZ cont3, por2
        MOV cont3, #3
por1:
        SETB senal
        djnz cont3, por1


        RET
```

```
END
```

```
                                        Luz y alarma
;*********************************************/
;                                             /
;Programa que controla la alarma sonora       /
; y la luz del cuarto                         /
;                                             /
;                                             /
;*********************************************/


cont1   EQU     R2   ;registro útil para retardo 10ms
cont2   EQU     R3   ;registro útil para retardo 10ms
cont3   equ     R4
cont4   EQU     R5
cont5   EQU     R6

;entradas
sen_foco        EQU     P0.0    ;estamos checando cuando se va a activar el
foco, con 0 off 1 on
sen_alarma      EQU     P0.3    ;activacion de la alarma, se activa al detectar
un 1, con 0 no hace nada

;salidas
enc_foco        EQU     P1.0
enc_alarma      EQU     P1.1

;en este programa siempre va a checar primero el foco y después la alarma
;enciendo el foco con "1", cuando tengo 1 en la entrada del foco lo prendo y con
0 lo apago

ORG 0

        jmp main
main:
        MOV     SP,#70H

esp_foco:
        JNB     sen_foco,apagado
                ;foco activo en 1
                ;se activo el foco, voy a prenderlo con 0 en enc_foco

        CLR     enc_foco        ;prendo el led con 0
        jmp     alarma
        JMP     esp_foco

apagado:
        SETB enc_foco  ;prendo el led con 0
        jmp alarma
        JMP esp_foco

alarma:
        JNB sen_alarma, no_func
        call ciclo_alar

no_func:
        JMP esp_foco

;subrutinas que se mandan llamar

ciclo_alar:
        mov cont5,#2       ;con esto se hacen dos bips
ciclo10:

        mov cont3,#50
ciclo:
        setb enc_alarma
        call retardo1ms
        clr enc_alarma
        call retardo1ms
```

```
        djnz cont3,ciclo
        mov cont4,#20

ciclo3:
        clr enc_alarma
        call retardo10ms
        djnz cont4,ciclo3
        djnz cont5,ciclo10
        RET

;SUBRUTINA QUE GENERA LOS 10ms

retardo1ms:
        MOV cont1,#5

ciclo2:
        MOV cont2, #97
ciclo1:
        DJNZ cont2, ciclo1
        DJNZ cont1, ciclo2
        RET

retardo10ms:
        MOV cont1, #25
ciclo2:
        MOV cont2, #200
ciclo1:
        DJNZ cont2, ciclo1
        DJNZ cont1, ciclo2
        RET


;FIN  SUBRUTINAS


END
```

DATA SHEET



2N2222; 2N2222A
NPN switching transistors

Product specification
Supersedes data of September 1994
File under Discrete Semiconductors, SC04

1997 May 29

Philips Semiconductors

PHILIPS

NPN switching transistors 2N2222; 2N2222A

FEATURES

- High current (max. 800 mA)
- Low voltage (max. 40 V).

APPLICATIONS

- Linear amplification and switching.

DESCRIPTION

NPN switching transistor in a TO-18 metal package.
PNP complement: 2N2907A.

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector, connected to case

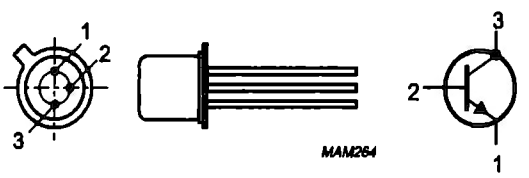


Fig.1 Simplified outline (TO-18) and symbol.

QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	2N2222		–	60	V
	2N2222A		–	75	V
V_{CEO}	collector-emitter voltage	open base			
	2N2222		–	30	V
	2N2222A		–	40	V
I_C	collector current (DC)		–	800	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
h_{FE}	DC current gain	I_C = 10 mA; V_{CE} = 10 V	75	–	
f_T	transition frequency	I_C = 20 mA; V_{CE} = 20 V; f = 100 MHz			
	2N2222		250	–	MHz
	2N2222A		300	–	MHz
t_{off}	turn-off time	I_{Con} = 150 mA; I_{Bon} = 15 mA; I_{Boff} = –15 mA	–	250	ns

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	2N2222		–	60	V
	2N2222A		–	75	V
V_{CEO}	collector-emitter voltage	open base			
	2N2222		–	30	V
	2N2222A		–	40	V
V_{EBO}	emitter-base voltage	open collector			
	2N2222		–	5	V
	2N2222A		–	6	V
I_C	collector current (DC)		–	800	mA
I_{CM}	peak collector current		–	800	mA
I_{BM}	peak base current		–	200	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
		$T_{case} \leq 25$ °C	–	1.2	W
T_{stg}	storage temperature		–65	+150	°C
T_j	junction temperature		–	200	°C
T_{amb}	operating ambient temperature		–65	+150	°C

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j\text{-}a}$	thermal resistance from junction to ambient	in free air	350	K/W
$R_{th\ j\text{-}c}$	thermal resistance from junction to case		146	K/W

CHARACTERISTICS

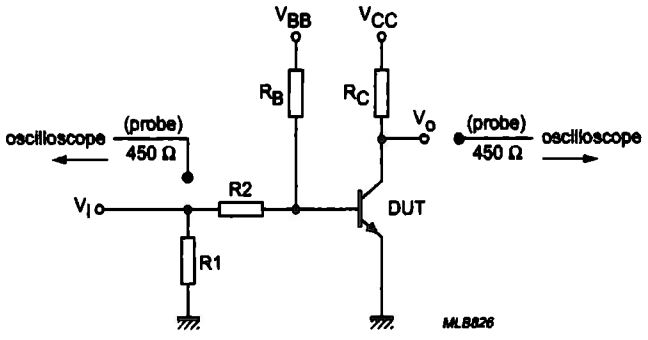
T_j = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
I_{CBO}	collector cut-off current				
	2N2222	I_E = 0; V_{CB} = 50 V	–	10	nA
		I_E = 0; V_{CB} = 50 V; T_{amb} = 150 °C	–	10	µA
I_{CBO}	collector cut-off current				
	2N2222A	I_E = 0; V_{CB} = 60 V	–	10	nA
		I_E = 0; V_{CB} = 60 V; T_{amb} = 150 °C	–	10	µA
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = 3 V	–	10	nA
h_{FE}	DC current gain	I_C = 0.1 mA; V_{CE} = 10 V	35	–	
		I_C = 1 mA; V_{CE} = 10 V	50	–	
		I_C = 10 mA; V_{CE} = 10 V	75	–	
		I_C = 150 mA; V_{CE} = 1 V; note 1	50	–	
		I_C = 150 mA; V_{CE} = 10 V; note 1	100	300	
h_{FE}	DC current gain	I_C = 10 mA; V_{CE} = 10 V; T_{amb} = –55 °C			
	2N2222A		35	–	
h_{FE}	DC current gain	I_C = 500 mA; V_{CE} = 10 V; note 1			
	2N2222		30	–	
	2N2222A		40	–	
V_{CEsat}	collector-emitter saturation voltage				
	2N2222	I_C = 150 mA; I_B = 15 mA; note 1	–	400	mV
		I_C = 500 mA; I_B = 50 mA; note 1	–	1.6	V
V_{CEsat}	collector-emitter saturation voltage				
	2N2222A	I_C = 150 mA; I_B = 15 mA; note 1	–	300	mV
		I_C = 500 mA; I_B = 50 mA; note 1	–	1	V
V_{BEsat}	base-emitter saturation voltage				
	2N2222	I_C = 150 mA; I_B = 15 mA; note 1	–	1.3	V
		I_C = 500 mA; I_B = 50 mA; note 1	–	2.6	V
V_{BEsat}	base-emitter saturation voltage				
	2N2222A	I_C = 150 mA; I_B = 15 mA; note 1	0.6	1.2	V
		I_C = 500 mA; I_B = 50 mA; note 1	–	2	V
C_c	collector capacitance	I_E = i_e = 0; V_{CB} = 10 V; f = 1 MHz	–	8	pF
C_e	emitter capacitance	I_C = i_c = 0; V_{EB} = 500 mV; f = 1 MHz			
	2N2222A		–	25	pF
f_T	transition frequency	I_C = 20 mA; V_{CE} = 20 V; f = 100 MHz			
	2N2222		250	–	MHz
	2N2222A		300	–	MHz
F	noise figure	I_C = 200 µA; V_{CE} = 5 V; R_S = 2 kΩ;			
	2N2222A	f = 1 kHz; B = 200 Hz	–	4	dB

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
Switching times (between 10% and 90% levels); see Fig.2					
t_{on}	turn-on time	I_{Con} = 150 mA; I_{Bon} = 15 mA; I_{Boff} = −15 mA	–	35	ns
t_d	delay time		–	10	ns
t_r	rise time		–	25	ns
t_{off}	turn-off time		–	250	ns
t_s	storage time		–	200	ns
t_f	fall time		–	60	ns

Note

1. Pulse test: $t_p \leq 300\ \mu s$; $\delta \leq 0.02$.

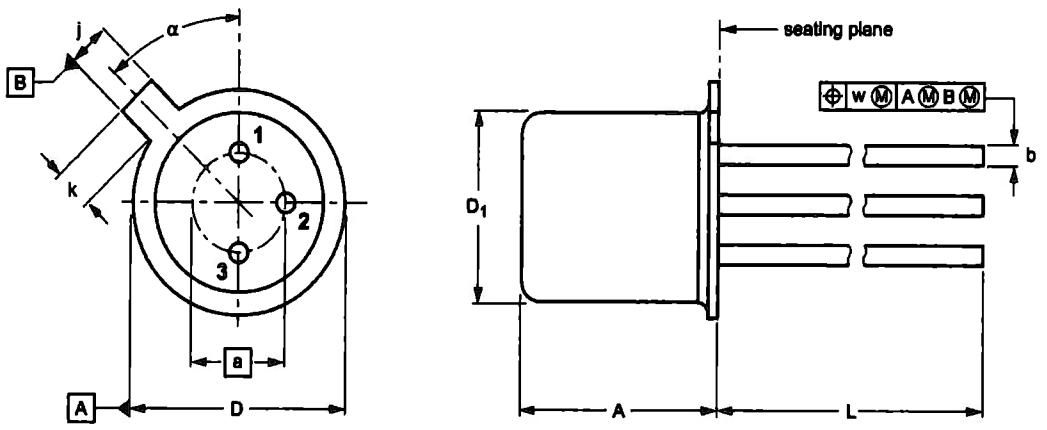


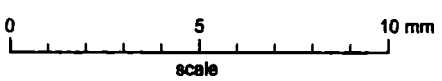
V_I = 9.5 V; T = 500 μs; t_p = 10 μs; $t_r = t_f \leq 3$ ns.
R1 = 68 Ω; R2 = 325 Ω; R_B = 325 Ω; R_C = 160 Ω.
V_{BB} = −3.5 V; V_{CC} = 29.5 V.
Oscilloscope input impedance Z_i = 50 Ω.

Fig.2 Test circuit for switching times.

PACKAGE OUTLINE

Metal-can cylindrical single-ended package; 3 leads **SOT18/13**





DIMENSIONS (millimetre dimensions are derived from the original inch dimensions)

UNIT	A	a	b	D	D_1	j	k	L	w	α
mm	5.31 4.74	2.54	0.47 0.41	5.45 5.30	4.70 4.55	1.03 0.94	1.1 0.9	15.0 12.7	0.40	45°

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT18/13	B11/C7 type 3	TO-18				97-04-18

DEFINITIONS

Data sheet status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application Information	
Where application information is given, it is advisory and does not form part of the specification.	

LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.

Philips Semiconductors – a worldwide company

Argentina: see South America

Australia: 34 Waterloo Road, NORTH RYDE, NSW 2113,
Tel. +61 2 9805 4455, Fax. +61 2 9805 4466

Austria: Computerstr. 6, A-1101 WIEN, P.O. Box 213,
Tel. +43 1 60 101, Fax. +43 1 60 101 1210

Belarus: Hotel Minsk Business Center, Bld. 3, r. 1211, Volodarski Str. 6,
220050 MINSK, Tel. +375 172 200 733, Fax. +375 172 200 773

Belgium: see The Netherlands

Brazil: see South America

Bulgaria: Philips Bulgaria Ltd., Energoproject, 15th floor,
51 James Bourchier Blvd., 1407 SOFIA,
Tel. +359 2 689 211, Fax. +359 2 689 102

Canada: PHILIPS SEMICONDUCTORS/COMPONENTS,
Tel. +1 800 234 7381

China/Hong Kong: 501 Hong Kong Industrial Technology Centre,
72 Tat Chee Avenue, Kowloon Tong, HONG KONG,
Tel. +852 2319 7888, Fax. +852 2319 7700

Colombia: see South America

Czech Republic: see Austria

Denmark: Prags Boulevard 80, PB 1919, DK-2300 COPENHAGEN S,
Tel. +45 32 88 2636, Fax. +45 31 57 0044

Finland: Sinikalliontie 3, FIN-02630 ESPOO,
Tel. +358 9 615800, Fax. +358 9 61580920

France: 4 Rue du Port-aux-Vins, BP317, 92156 SURESNES Cedex,
Tel. +33 1 40 99 6161, Fax. +33 1 40 99 6427

Germany: Hammerbrookstraße 69, D-20097 HAMBURG,
Tel. +49 40 23 53 60, Fax. +49 40 23 536 300

Greece: No. 15, 25th March Street, GR 17778 TAVROS/ATHENS,
Tel. +30 1 4894 339/239, Fax. +30 1 4814 240

Hungary: see Austria

India: Philips INDIA Ltd, Shivsagar Estate, A Block, Dr. Annie Besant Rd.
Worli, MUMBAI 400 018, Tel. +91 22 4938 541, Fax. +91 22 4938 722

Indonesia: see Singapore

Ireland: Newstead, Clonskeagh, DUBLIN 14,
Tel. +353 1 7640 000, Fax. +353 1 7640 200

Israel: RAPAC Electronics, 7 Kehilat Saloniki St, PO Box 18053,
TEL AVIV 61180, Tel. +972 3 645 0444, Fax. +972 3 649 1007

Italy: PHILIPS SEMICONDUCTORS, Piazza IV Novembre 3,
20124 MILANO, Tel. +39 2 6752 2531, Fax. +39 2 6752 2557

Japan: Philips Bldg 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108,
Tel. +81 3 3740 5130, Fax. +81 3 3740 5077

Korea: Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL,
Tel. +82 2 709 1412, Fax. +82 2 709 1415

Malaysia: No. 76 Jalan Universiti, 46200 PETALING JAYA, SELANGOR,
Tel. +60 3 750 5214, Fax. +60 3 757 4880

Mexico: 5900 Gateway East, Suite 200, EL PASO, TEXAS 79905,
Tel. +9-5 800 234 7381

Middle East: see Italy

Netherlands: Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,
Tel. +31 40 27 82785, Fax. +31 40 27 88399

New Zealand: 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,
Tel. +64 9 849 4160, Fax. +64 9 849 7811

Norway: Box 1, Manglerud 0612, OSLO,
Tel. +47 22 74 8000, Fax. +47 22 74 8341

Philippines: Philips Semiconductors Philippines Inc.,
106 Valero St. Salcedo Village, P.O. Box 2108 MCC, MAKATI,
Metro MANILA, Tel. +63 2 816 6380, Fax. +63 2 817 3474

Poland: Ul. Lukiska 10, PL 04-123 WARSZAWA,
Tel. +48 22 612 2831, Fax. +48 22 612 2327

Portugal: see Spain

Romania: see Italy

Russia: Philips Russia, Ul. Usatcheva 35A, 119048 MOSCOW,
Tel. +7 095 755 6918, Fax. +7 095 755 6919

Singapore: Lorong 1, Toa Payoh, SINGAPORE 1231,
Tel. +65 350 2538, Fax. +65 251 6500

Slovakia: see Austria

Slovenia: see Italy

South Africa: S.A. PHILIPS Pty Ltd., 195-215 Main Road Martindale,
2092 JOHANNESBURG, P.O. Box 7430 Johannesburg 2000,
Tel. +27 11 470 5911, Fax. +27 11 470 5494

South America: Rua do Rocio 220, 5th floor, Suite 51,
04552-903 São Paulo, SÃO PAULO - SP, Brazil,
Tel. +55 11 821 2333, Fax. +55 11 829 1849

Spain: Balmes 22, 08007 BARCELONA,
Tel. +34 3 301 6312, Fax. +34 3 301 4107

Sweden: Kottbygatan 7, Akalla, S-16485 STOCKHOLM,
Tel. +46 8 632 2000, Fax. +46 8 632 2745

Switzerland: Allmendstrasse 140, CH-8027 ZÜRICH,
Tel. +41 1 488 2686, Fax. +41 1 481 7730

Taiwan: Philips Semiconductors, 6F, No. 96, Chien Kuo N. Rd., Sec. 1,
TAIPEI, Taiwan Tel. +886 2 2134 2865, Fax. +886 2 2134 2874

Thailand: PHILIPS ELECTRONICS (THAILAND) Ltd.,
209/2 Sanpavuth-Bangna Road Prakanong, BANGKOK 10260,
Tel. +66 2 745 4090, Fax. +66 2 398 0793

Turkey: Talatpasa Cad. No. 5, 80640 GÜLTEPE/ISTANBUL,
Tel. +90 212 279 2770, Fax. +90 212 282 6707

Ukraine: PHILIPS UKRAINE, 4 Patrice Lumumba str., Building B, Floor 7,
252042 KIEV, Tel. +380 44 264 2776, Fax. +380 44 268 0461

United Kingdom: Philips Semiconductors Ltd., 276 Bath Road, Hayes,
MIDDLESEX UB3 5BX, Tel. +44 181 730 5000, Fax. +44 181 754 8421

United States: 811 East Arques Avenue, SUNNYVALE, CA 94088-3409,
Tel. +1 800 234 7381

Uruguay: see South America

Vietnam: see Singapore

Yugoslavia: PHILIPS, Trg N. Pasica 5/v, 11000 BEOGRAD,
Tel. +381 11 625 344, Fax.+381 11 635 777

For all other countries apply to: Philips Semiconductors, Marketing & Sales Communications,
Building BE-p, P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands, Fax. +31 40 27 24825

Internet: http://www.semiconductors.philips.com

Printed in The Netherlands 117047/00/02/pp8 Date of release: 1997 May 29 Document order number: 9397 750 02161

Let's make things better.

Philips PHILIPS

FAIRCHILD
SEMICONDUCTOR®

BC556/557/558/559/560

Switching and Amplifier

- High Voltage: BC556, V_{CEO}= -65V
- Low Noise: BC559, BC560
- Complement to BC546 ... BC 550

TO-92

1

1. Collector 2. Base 3. Emitter

PNP Epitaxial Silicon Transistor

Absolute Maximum Ratings T_a=25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage		
	: BC556	-80	V
	: BC557/560	-50	V
	: BC558/559	-30	V
V_{CEO}	Collector-Emitter Voltage		
	: BC556	-65	V
	: BC557/560	-45	V
	: BC558/559	-30	V
V_{EBO}	Emitter-Base Voltage	-5	V
I_C	Collector Current (DC)	-100	mA
P_C	Collector Power Dissipation	500	mW
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	-65 ~ 150	°C

Electrical Characteristics T_a=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
I_{CBO}	Collector Cut-off Current	V_{CB}= -30V, I_E=0			-15	nA
h_{FE}	DC Current Gain	V_{CE}= -5V, I_C=2mA	110		800	
V_{CE} (sat)	Collector-Emitter Saturation Voltage	I_C= -10mA, I_B= -0.5mA		-90	-300	mV
		I_C= -100mA, I_B= -5mA		-250	-650	mV
V_{BE} (sat)	Collector-Base Saturation Voltage	I_C= -10mA, I_B= -0.5mA		-700		mV
		I_C= -100mA, I_B= -5mA		-900		mV
V_{BE} (on)	Base-Emitter On Voltage	V_{CE}= -5V, I_C= -2mA	-600	-660	-750	mV
		V_{CE}= -5V, I_C= -10mA			-800	mV
f_T	Current Gain Bandwidth Product	V_{CE}= -5V, I_C= -10mA, f=10MHz		150		MHz
C_{ob}	Output Capacitance	V_{CB}= -10V, I_E=0, f=1MHz			6	pF
NF	Noise Figure : BC556/557/558	V_{CE}= -5V, I_C= -200μA		2	10	dB
	: BC559/560	f=1KHz, R_G=2KΩ		1	4	dB
	: BC559	V_{CE}= -5V, I_C= -200μA		1.2	4	dB
	: BC560	R_G=2KΩ, f=30~15000MHz		1.2	2	dB

h_{FE} Classification

Classification	A	B	C
h_{FE}	110 ~ 220	200 ~ 450	420 ~ 800

Typical Characteristics

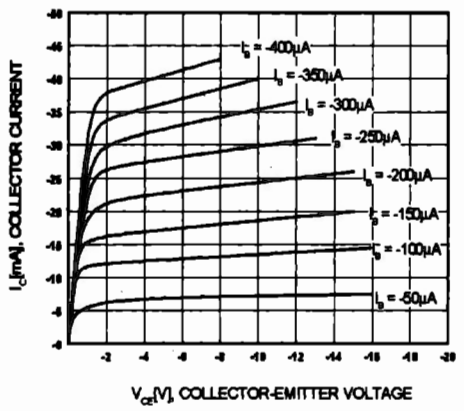


Figure 1. Static Characteristic

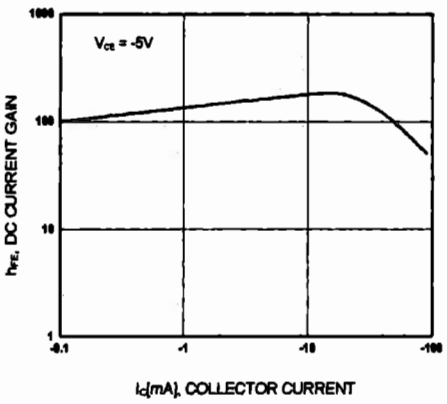


Figure 2. DC current Gain

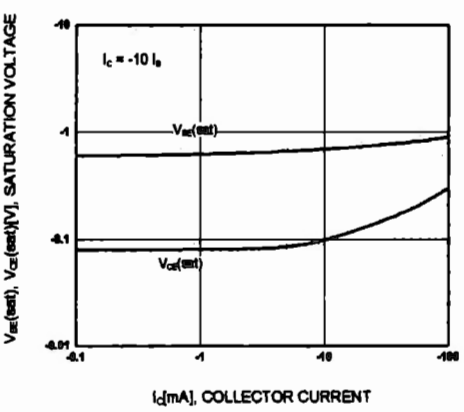


Figure 3. Base-Emitter Saturation Voltage Collector-Emitter Saturation Voltage

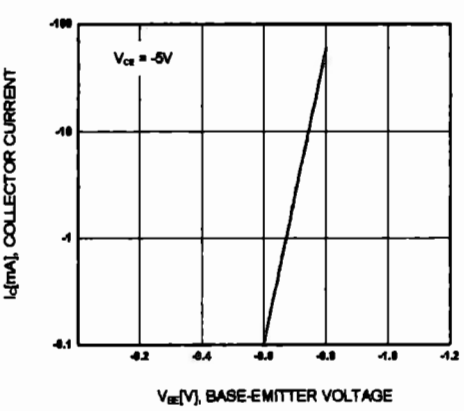


Figure 4. Base-Emitter On Voltage

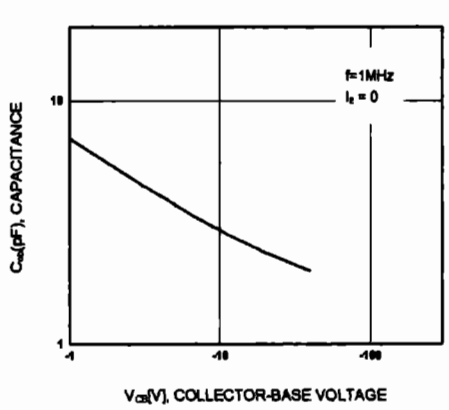


Figure 5. Collector Output Capacitance

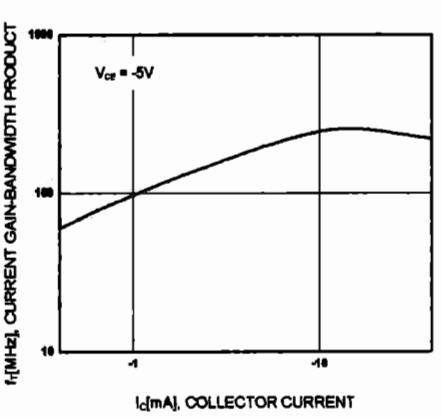
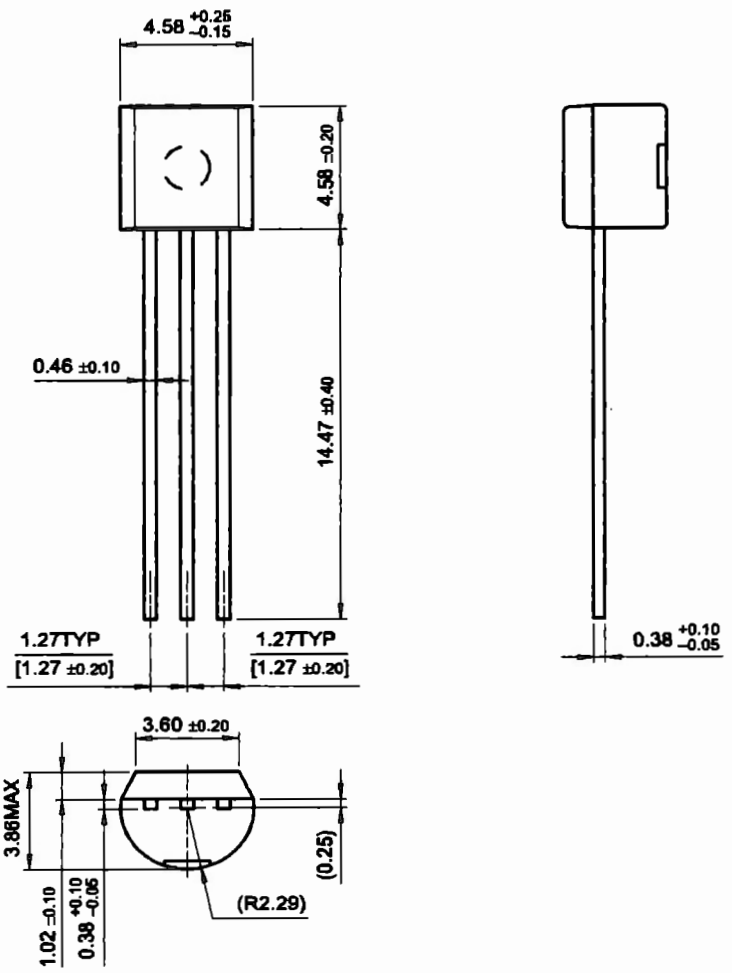


Figure 6. Current Gain Bandwidth Product

Package Dimensions

TO-92



Dimensions in Millimeters

TRADEMARKS

The following are registered and unregistered trademarks Fairchild Semiconductor owns or is authorized to use and is not intended to be an exhaustive list of all such trademarks.

ACEx™	FACT™	ImpliedDisconnect™	PACMAN™	SPM™
ActiveArray™	FACT Quiet series™	ISOPLANAR™	POP™	Stealth™
Bottomless™	FAST®	LittleFET™	Power247™	SuperSOT™-3
CoolFET™	FASTr™	MicroFET™	PowerTrench®	SuperSOT™-6
CROSSVOLT™	FRFET™	MicroPak™	QFET™	SuperSOT™-8
DOME™	GlobalOptoisolator™	MICROWIRE™	QS™	SyncFET™
EcoSPARK™	GTO™	MSX™	QT Optoelectronics™	TinyLogic™
E^{2}CMOS™	HiSeC™	MSXPro™	Quiet Series™	TruTranslation™
EnSigna™	I^{2}C™	OCX™	RapidConfigure™	UHC™
Across the board. Around the world.™		OCXPro™	RapidConnect™	UltraFET®
The Power Franchise™		OPTOLOGIC®	SILENT SWITCHER®	VCX™
Programmable Active Droop™		OPTOPLANAR™	SMART START™	

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.

As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, or (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

PRODUCT STATUS DEFINITIONS

Definition of Terms

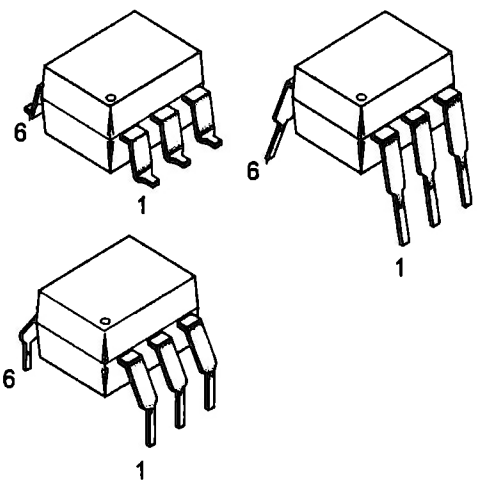
Datasheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This datasheet contains preliminary data, and supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
No Identification Needed	Full Production	This datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice in order to improve design.
Obsolete	Not In Production	This datasheet contains specifications on a product that has been discontinued by Fairchild semiconductor. The datasheet is printed for reference information only.

FAIRCHILD
SEMICONDUCTOR®

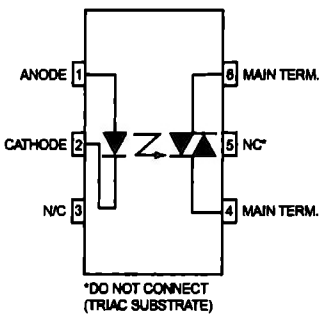
6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

MOC3010M MOC3011M MOC3012M MOC3020M MOC3021M MOC3022M MOC3023M

PACKAGE



SCHEMATIC



DESCRIPTION

The MOC301XM and MOC302XM series are optically isolated triac driver devices. These devices contain a GaAs infrared emitting diode and a light activated silicon bilateral switch, which functions like a triac. They are designed for interfacing between electronic controls and power triacs to control resistive and inductive loads for 115 VAC operations.

FEATURES

- Excellent I_{FT} stability—IR emitting diode has low degradation
- High isolation voltage—minimum 5300 VAC RMS
- Underwriters Laboratory (UL) recognized—File #E90700
- Peak blocking voltage
  - 250V-MOC301XM
  - 400V-MOC302XM
- VDE recognized (File #94766)
  - Ordering option V (e.g. MOC3023VM)

APPLICATIONS

- Industrial controls
- Traffic lights
- Vending machines
- Solid state relay
- Lamp ballasts
- Solenoid/valve controls
- Static AC power switch
- Incandescent lamp dimmers
- Motor control

FAIRCHILD
SEMICONDUCTOR®

6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

MOC3010M MOC3011M MOC3012M MOC3020M MOC3021M MOC3022M MOC3023M

ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)				
Parameters	**Symbol**	**Device**	**Value**	**Units**
TOTAL DEVICE				
Storage Temperature	T_{STG}	All	-40 to +150	°C
Operating Temperature	T_{OPR}	All	-40 to +85	°C
Lead Solder Temperature	T_{SOL}	All	260 for 10 sec	°C
Junction Temperature Range	T_J	All	-40 to +100	°C
Isolation Surge Voltage(1) (peak AC voltage, 60Hz, 1 sec duration)	V_{ISO}	All	7500	Vac(pk)
Total Device Power Dissipation @ 25°C	P_D	All	330	mW
Derate above 25°C			4.4	mW/°C
EMITTER				
Continuous Forward Current	I_F	All	60	mA
Reverse Voltage	V_R	All	3	V
Total Power Dissipation 25°C Ambient	P_D	All	100	mW
Derate above 25°C			1.33	mW/°C
DETECTOR				
Off-State Output Terminal Voltage	V_{DRM}	MOC3010M/1M/2M MOC3020M/1M/2M/3M	250 400	V
Peak Repetitive Surge Current (PW = 1 ms, 120 pps)	I_{TSM}	All	1	A
Total Power Dissipation @ 25°C Ambient	P_D	All	300	mW
Derate above 25°C			4	mW/°C

Note

1. Isolation surge voltage, V_{ISO}, is an internal device dielectric breakdown rating. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

ELECTRICAL CHARACTERISTICS (T_A = 25°C Unless otherwise specified)

INDIVIDUAL COMPONENT CHARACTERISTICS

Parameters	Test Conditions	Symbol	Device	Min	Typ	Max	Units
EMITTER							
Input Forward Voltage	I_F = 10 mA	V_F	All		1.15	1.5	V
Reverse Leakage Current	V_R = 3 V, T_A = 25°C	I_R	All		0.01	100	µA
DETECTOR							
Peak Blocking Current,Either Direction	Rated V_{DRM}, I_F = 0 (note 1)	I_{DRM}	All		10	100	nA
Peak On-State Voltage,Either Direction	I_{TM} = 100 mA peak, I_F = 0	V_{TM}	All		1.8	3	V

TRANSFER CHARACTERISTICS (T_A = 25°C Unless otherwise specified.)

DC Characteristics	Test Conditions	Symbol	Device	Min	Typ	Max	Units
LED Trigger Current	Voltage = 3V (note 3)	I_{FT}	MOC3020M			30	mA
			MOC3010M			15	
			MOC3021M				
			MOC3011M			10	
			MOC3022M				
			MOC3012M			5	
			MOC3023M				
Holding Current, Either Direction		I_H	All		100		µA

Note

1. Test voltage must be applied within dv/dt rating.
2. This is static dv/dt. See Figure 5 for test circuit. Commutating dv/dt is a function of the load-driving thyristor(s) only.
3. All devices are guaranteed to trigger at an I_F value less than or equal to max I_{FT}. Therefore, recommended operating I_F lies between max I_{FT} (30 mA for MOC3020M, 15 mA for MOC3010M and MOC3021M, 10 mA for MOC3011M and MOC3022M, 5 mA for MOC3012M and MOC3023M) and absolute max I_F (60 mA).

6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

Fig. 1 LED Forward Voltage vs. Forward Current

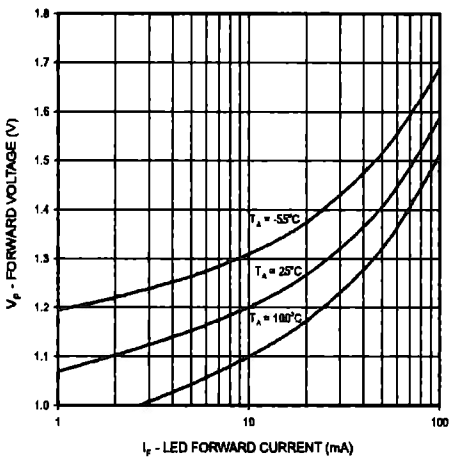


Fig. 2 On-State Characteristics

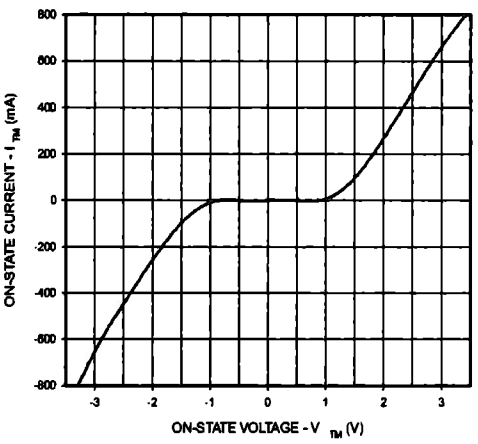


Fig. 3 Trigger Current vs. Ambient Temperature

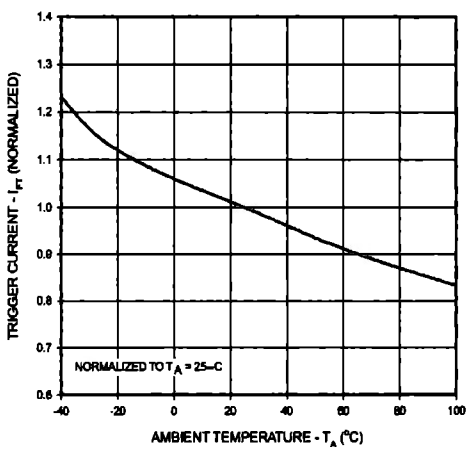


Fig. 4 LED Current Required to Trigger vs. LED Pulse Width

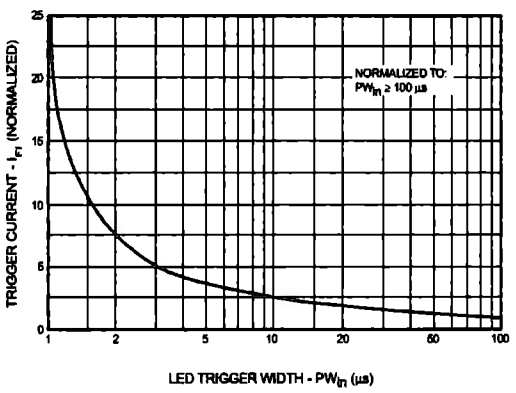


Fig. 5 dv/dt vs. Temperature

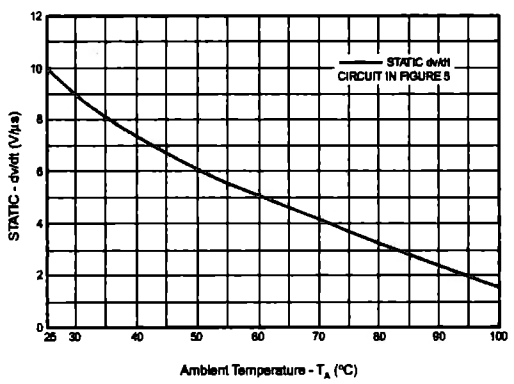
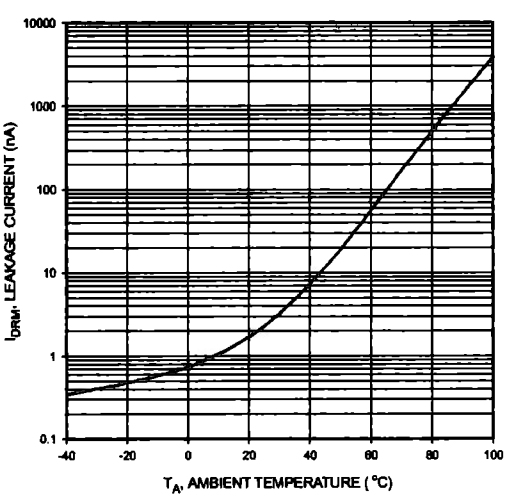
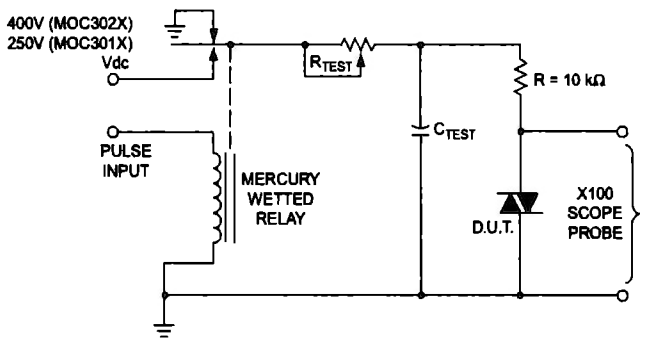


Fig. 6 Leakage Current, I_{DRM} vs. Temperature





1. The mercury wetted relay provides a high speed repeated pulse to the D.U.T.
2. 100x scope probes are used, to allow high speeds and voltages.
3. The worst-case condition for static dv/dt is established by triggering the D.U.T. with a normal LED input current, then removing the current. The variable R_{TEST} allows the dv/dt to be gradually increased until the D.U.T. continues to trigger in response to the applied voltage pulse, even after the LED current has been removed. The dv/dt is then decreased until the D.U.T. stops triggering. τ_{RC} is measured at this point and recorded.

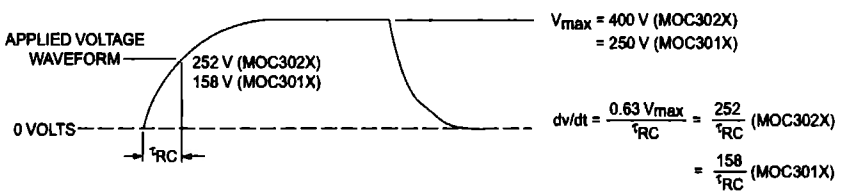


Figure 5. Static dv/dt Test Circuit

Note: This optoisolator should not be used to drive a load directly. It is intended to be a trigger device only.

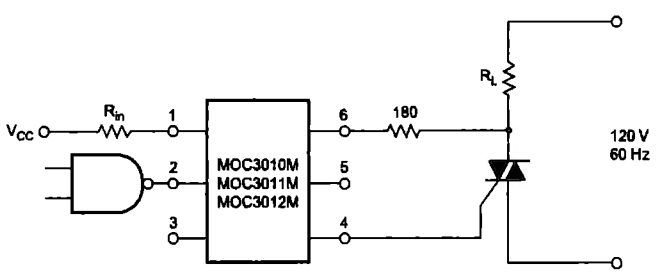


Figure 6. Resistive Load

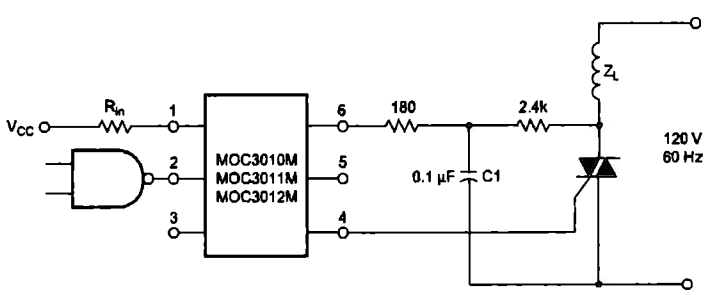


Figure 7. Inductive Load with Sensitive Gate Triac ($I_{GT} \leq 15$ mA)

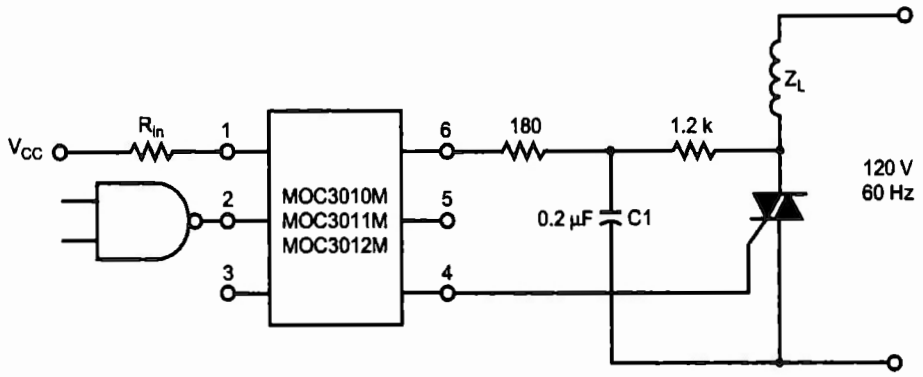
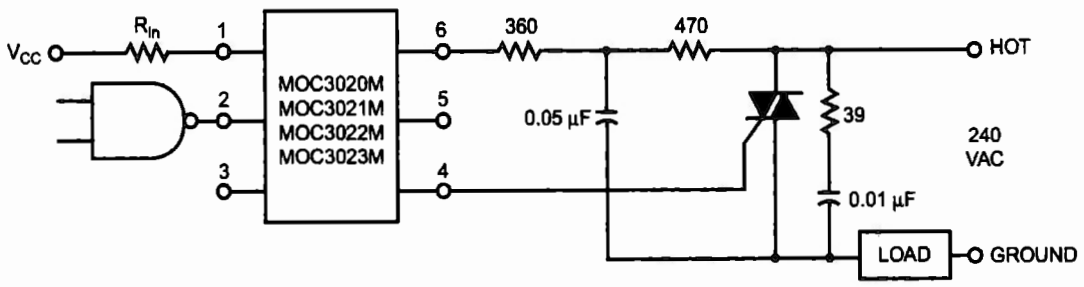


Figure 8. Inductive Load with Sensitive Gate Triac ($I_{GT} \leq 15$ mA)



In this circuit the "hot" side of the line is switched and the load connected to the cold or ground side.

The 39 ohm resistor and 0.01µF capacitor are for snubbing of the triac, and the 470 ohm resistor and 0.05 µF capacitor are for snubbing the coupler. These components may or may not be necessary depending upon the particular and load used.

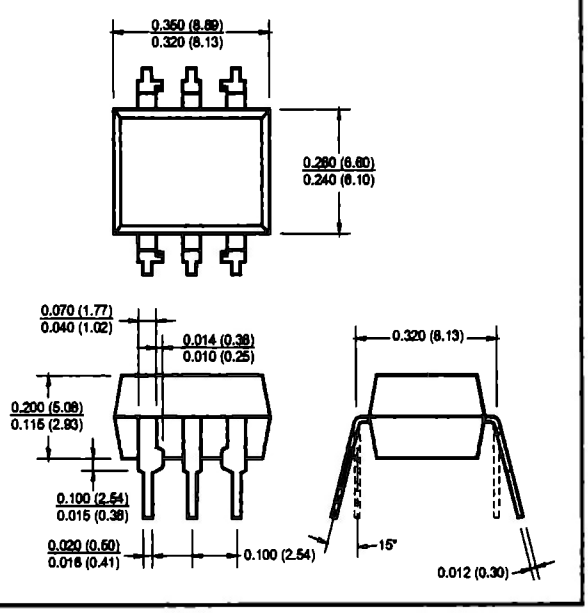
Figure 9. Typical Application Circuit

FAIRCHILD
SEMICONDUCTOR®

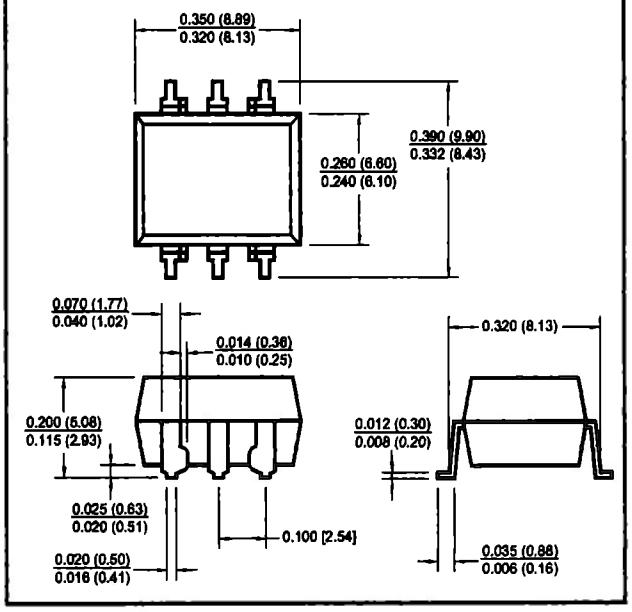
6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

MOC3010M MOC3011M MOC3012M MOC3020M MOC3021M MOC3022M MOC3023M

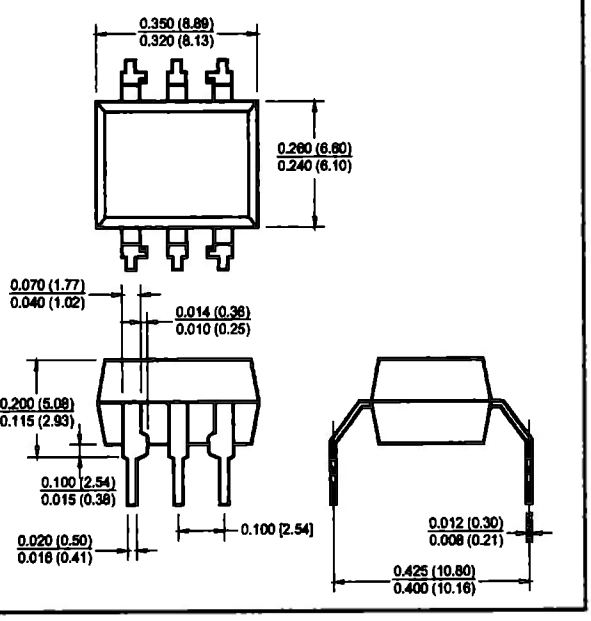
Package Dimensions (Through Hole)



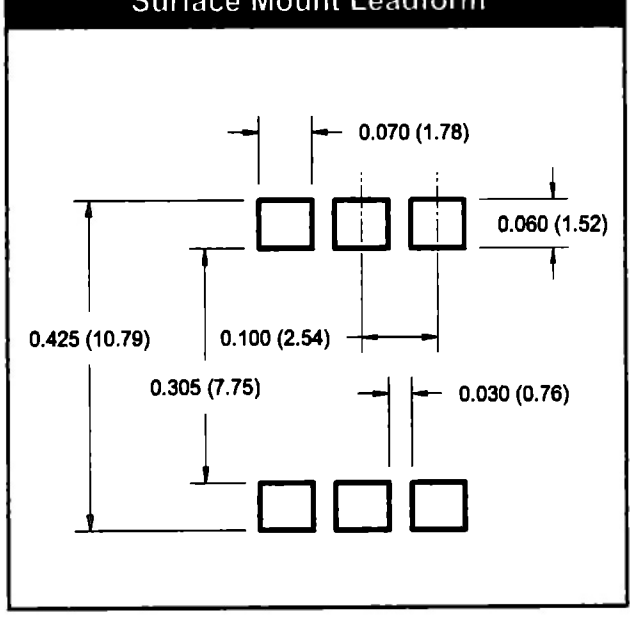
Package Dimensions (Surface Mount)



Package Dimensions (0.4" Lead Spacing)



Recommended Pad Layout for Surface Mount Leadform



NOTE
All dimensions are in inches (millimeters)

FAIRCHILD
SEMICONDUCTOR®

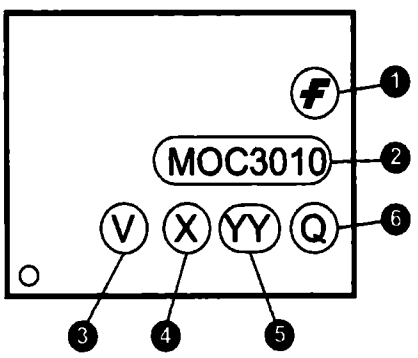
6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

MOC3010M MOC3011M MOC3012M MOC3020M MOC3021M MOC3022M MOC3023M

ORDERING INFORMATION

Option	Order Entry Identifier	Description
S	S	Surface Mount Lead Bend
SR2	SR2	Surface Mount; Tape and reel
T	T	0.4" Lead Spacing
V	V	VDE 0884
TV	TV	VDE 0884, 0.4" Lead Spacing
SV	SV	VDE 0884, Surface Mount
SR2V	SR2V	VDE 0884, Surface Mount, Tape & Reel

MARKING INFORMATION

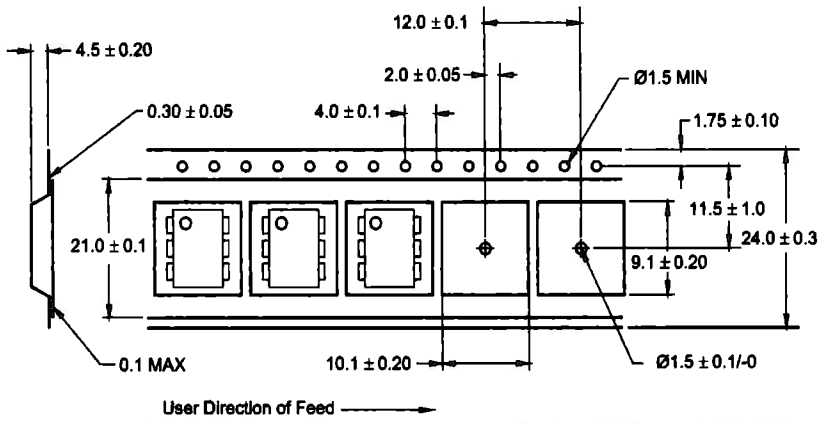


Definitions	
1	Fairchild logo
2	Device number
3	VDE mark (Note: Only appears on parts ordered with VDE option – See order entry table)
4	One digit year code, e.g., '3'
5	Two digit work week ranging from '01' to '53'
6	Assembly package code

*Note – Parts that do not have the 'V' option (see definition 3 above) that are marked with date code '325' or earlier are marked in portrait format.

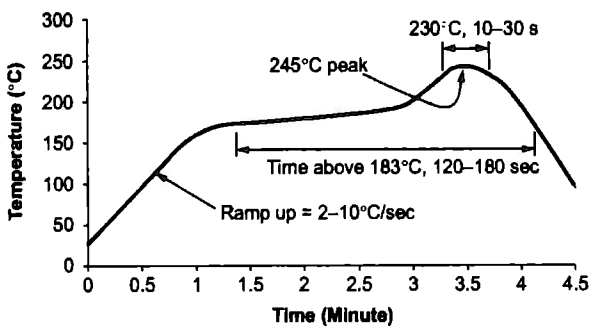
6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

Carrier Tape Specifications



NOTE
All dimensions are in inches (millimeters)

Reflow Profile (White Package, -M Suffix)



- Peak reflow temperature: 245°C (package surface temperature)
- Time of temperature higher than 183°C for 120–180 seconds
- One time soldering reflow is recommended

FAIRCHILD
SEMICONDUCTOR®

6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK)

MOC3010M MOC3011M MOC3012M MOC3020M MOC3021M MOC3022M MOC3023M

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

```
#include <windows.h>
#include <stdio.h>


void PressKey(char k){
        keybd_event( k, 0x45, KEYEVENTF_EXTENDEDKEY | 0, 0 );
        keybd_event( k, 0x45, KEYEVENTF_EXTENDEDKEY | KEYEVENTF_KEYUP, 0);
}

void main()
{
    HWND hwnd_unefon;
        STARTUPINFO si;
        PROCESS_INFORMATION pi;
    char buffer[256];
    int i;
    char tel[]="5516455372";
    char sms[]="AYUDA POR FAVOR, GRACIAS ";

    ZeroMemory( &si, sizeof(si) );
        si.cb = sizeof(si);
        ZeroMemory( &pi, sizeof(pi) );

    // Comienza el proceso para abrir pagina de internet.
        if( !CreateProcess( NULL,
        "\"C:\\Archivos de programa\\Internet Explorer\\iexplore.exe\" http://www.unefon. ↙
    com.mx/sms/index.jsp", // Linea de comando .
        NULL,
        NULL,
        FALSE,
        0,
        NULL,
        NULL,
        &si,
        &pi )
    )
    {
        printf( "CreateProcess failed (%d).\n", GetLastError() );
        return;
    }

    Sleep(10000);

    hwnd_unefon = GetForegroundWindow();
    GetWindowText(hwnd_unefon, buffer, 256);
    buffer[6]='\0';
    if (strcmp(buffer,"UNEFON") != 0)
        return;

    PressKey(VK_TAB);
    PressKey(VK_TAB);

    // Genera las teclas del número del telefono
    i = 0;
    while (tel[i] != '\0')
    {
        PressKey(tel[i]);
        i++;
    }

    PressKey(VK_TAB);

    // Genera las teclas del mensaje
    i = 0;
    while (sms[i] != '\0')
    {
        PressKey(sms[i]);
        i++;
    }


    PressKey(VK_TAB);
    PressKey(VK_TAB);
```

```
    // Se uasa para cerrar la ventana

    PressKey(VK_RETURN);
    Sleep(10000);
    PressKey(VK_TAB);
    PressKey(VK_TAB);
    PressKey(VK_TAB);
    PressKey(VK_RETURN);
    PressKey(VK_RETURN);

    // Cierra los procesos.
    CloseHandle( pi.hProcess );
    CloseHandle( pi.hThread );
}
```