Universidad Politécnica de Cartagena

industriales
etsii UPCT

PROYECTO FIN DE CARRERA

“Diseño y construcción de una maqueta para el control semafórico con Arduino”

Titulación: INGIENERÍA TÉCNICA INDUSTRIAL, ESP. ELECTRÓNICA IND.

Alumno/a: FRANCISCO JAVIER TOLEDANO MORENO.

Director/a/s: MIGUEL ALMONACID KROEGER.

Cartagena, 20 de Marzo de 2012

Agradecimientos

Me gustaría agradecer el apoyo de mi familia y amigos, quienes han creído siempre en mí y en los objetivos que lucho por cumplir, haciendo posible que me encuentre escribiendo estas líneas.

Como no, expresar mi más sincero agradecimiento hacia el director de este Proyecto Fin de Carrera, el Dr. Miguel Almonacid Kroeger, el cual me ha prestado su ayuda incondicional y guiado en la realización de dicho proyecto.

Por último, mencionar al Departamento de Ingeniería de Sistemas y Automática y la aportación de su personal.

Índice

Capítulo 1

MOTIVACIÓN Y OBJETIVOS.

1.1 MOTIVACIÓN

El principal motivo de la realización del presente Proyecto Fin de Carrera radica en el profundo interés en las capacidades de Arduino como herramienta de control electrónico. El manejo de Arduino implica una gran reducción de tiempo en diseño electrónico y en la programación de su microcontrolador.

Sus características hacen de él una oportunidad de negocio en cuanto a su implementación en los sectores de la automatización y las comunicaciones industriales, a fin de una futura dedicación laboral en el marco de estos sectores industriales.

Además, el desarrollo del Proyecto Fin de Carrera concluirá con la obtención del título en Ingeniería Técnica Industrial, especialidad en Electrónica Industrial otorgado por la Universidad Politécnica de Cartagena.

1.2 OBJETIVOS Y FASES DEL PROYECTO

El principal objetivo de este proyecto es la familiarización con Arduino, su implementación en el control de grupos de semáforos y la construcción de una maqueta que ejemplifique un sistema semafórico real.

Las fases del proyecto son las siguientes:

- Estudio en profundidad de la placa Arduino Uno.
- Diseño de un Grafcet para el control de tres semáforos de coches, uno peatonal con pulsador de puesta en marcha y display de cuenta atrás. Además de un sensor de barrera infrarrojo para la detección de coches por una de las vías con poco tránsito.
- Primer programa: control de un semáforo.
- Segundo programa: control de tres semáforos en secuencia.
- Tercer programa: control de tres semáforos de coches y un peatonal.
- Cuarto programa: añadir un display para la cuenta atrás del semáforo de peatones.
- Quinto programa: incluir el uso de la barrera de infrarrojos en el programa.
- Diseño del circuito electrónico a controlar con Arduino.
- Diseño de la placa de circuito impreso (PCB).
- Construcción y comprobación de la PCB.

Capítulo 2

ARDUINO UNO.

2.1 INTRODUCCIÓN

El proyecto está basado en la utilización del hardware y software de Arduino. Éste es una plataforma de prototipos electrónica de código abierto (**open-source**) basada en hardware y software flexibles y relativamente fáciles de usar. Sus fabricantes publicitan a Arduino como una herramienta pensada para artistas, diseñadores y para cualquier interesado en crear objetos o entornos interactivos.

Arduino consta de un microcontrolador **ATmega** de Armel. El hardware de Arduino no deja de ser una plataforma microcontroladora para computación física como otras muchas disponibles en el mercado. Las diferencias, en cambio, con otras tarjetas de programación radica en lo siguiente**: bajo coste,** su software se ejecuta en los sistemas operativos más extendidos **(Windows, Linus y Macintosh)**, incorpora **funciones específicas de Arduino** que simplifican la programación, **hardware modificable y extensible**, y **software de código abierto** para su reprogramación por cualquier programador que lo desee.

Hay múltiples versiones de la placa Arduino. Entre ellas se encuentran algunas muy interesantes como **Arduino BT** que contiene un módulo bluetooth para la comunicación y programación sin cables, **Arduino Mini** que es la placa más pequeña y **Arduino Serial** que usa RS232 como interfaz con el ordenador para programación y comunicación.

Arduino es una herramienta a tener en cuenta por su versatilidad y bajo coste en uso industrial. Es muy útil en aquellas situaciones en las que se necesita controlar un sistema o producto del que se van a fabricar un pequeño número de unidades. En esta situación el ingeniero no necesita emplearse en el diseño electrónico de la tarjeta de control del microcontrolador a utilizar, pues ya viene diseñada y lista para cargar tu programa.

Su uso se puede extender al control de procesos en industrias reducidas adaptando el valor de tensión de las entradas y salidas con el uso de etapas de optoacopladores. A la hora de usar un sensor nos permite su linealización interna dando lugar a la optimización de la recogida de datos y la consecuente reducción de coste en transductores electrónicos.

2.2 CARACTERÍSTICAS BÁSICAS DE ARDUINO

En este proyecto se utiliza la última versión de Arduino: **Arduino Uno**.

Figura 2.2.1 Frontal y reverso de la placa Arduino Uno

Esta placa tiene un tamaño de 74x53mm. Usa para la programación una conexión USB a través de la cual puede ser alimentada (5v). En caso de usar alimentación externa esta ha de ser de 9 voltios. No es necesaria en esta placa hacer la selección del tipo de alimentación pues esta se hace automáticamente. Arduino Uno consta de 14 entradas digitales y 6 analógicas que pueden usarse también como si fueran digitales. Además puedes alimentar tu circuito a 5 o 3.3 voltios a través de ella.

Características básicas:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

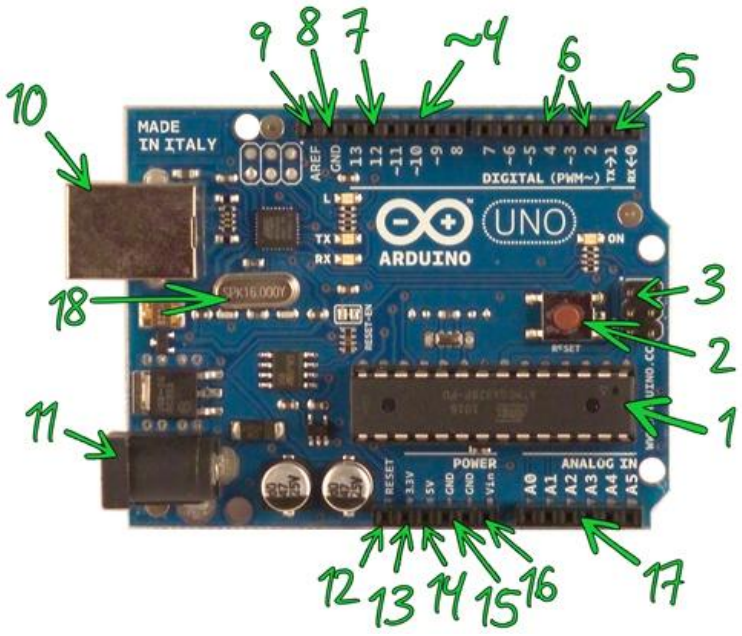


Figura 2.2.2 Identificación de elementos en Arduino Uno

1.- Microcontrolador ATmega328

Es un microcontrolador de la compañía Atmel que cuenta con 32KB de memoria flash, 2KB de memoria RAM y 1KB de memoria EEPROM. El microcontrolador puede ser utilizado como reemplazo del microcontrolador de las Freeduino o las Arduino Duemilanove o Diecimila o también puede utilizarse para realizar el montaje de una Arduino desde protoboard.

Características
- Voltaje de Operación: 5V
- Memoria Flash: 32 KB de los cuales 512 bytes son utilizados por el bootloader
- SRAM 2 KB
- EEPROM 1 KB
- Velocidad del Reloj 16 MHz
- Bootloader preinstalado

2.- Boton Reset

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

3.- ICSP

Conector para la programación ICSP (In Circuit Serial Programming, o Programación Serial en circuito). El ICSP es el sistema utilizado en los dispositivos PIC para programarlos sin necesidad de tener que retirar el chip del circuito del que forma parte.

4.- ~PWM

pines 3, 5, 6, 9, 10 y 11 provee de 8 bits de salida PWM con la función analogWrite (). La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

5.- Serie: 0 (RX) y 1 (TX)

Se utiliza para recibir (RX) y transmisión (TX) datos serie TTL. Estos pines están conectados a los pines correspondientes de la ATmega8U2 USB-to-TTL de chips de serie.

6.- Interrupciones externas

Pines 2 y 3 Estos pines pueden ser configurados para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor.

7.- SPI

10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK); Estos pines sirven de apoyo a la comunicación SPI con la biblioteca de SPI. El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj.

8.- GND

Pines de tierra. Abreviación de Ground que traducido al español es Tierra y en el contexto de la electrónica significa el común del circuito adonde se supone que existe 0 voltios.

9.- AREF

Tensión de referencia para las entradas analógicas. Se utiliza con analogReference ().

10.- USB

El Arduino Uno tiene una serie de facilidades para comunicarse con una computadora, Usando los canales de comunicación de esta serie a través de USB y aparece como un puerto COM virtual en el ordenador. Utiliza el estándar de los controladores USB COM, y no necesita ningún controlador externo. Sin embargo, en Windows es necesario un archivo .inf. El RX y TX LED de la placa parpadean cuando se transmiten datos a través del USB al chip serie y viceversa.

11.- Conector de alimentación

Plug hembra de 2.1mm para la conexión de alimentación en la placa.

12.- Reset

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

13.- 3.3 V

Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

14.- 5V

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

15.- GND

Pines de toma de tierra.

16.- VIN

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

17.- Analog In

El Uno tiene 6 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de tierra a 5 voltios, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference().

18.- Cristal.

2.3 ENTORNO DE PROGRAMACIÓN

Arduino uno sólo se puede programar usando la versión 0021 y posteriores de software. Debes descargarla e instalarla. El compilador es muy sencillo e intuitivo de utilizar. Lo más importante es seleccionar la placa a utilizar y el puerto de conexión.

El entorno de Desarrollo Arduino está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

En la figura 2.3.1 se puede ver la simplicidad de la interfaz del compilador usado por Arduino, la cual no cambia sea cual sea de las diferentes versiones de Arduino.

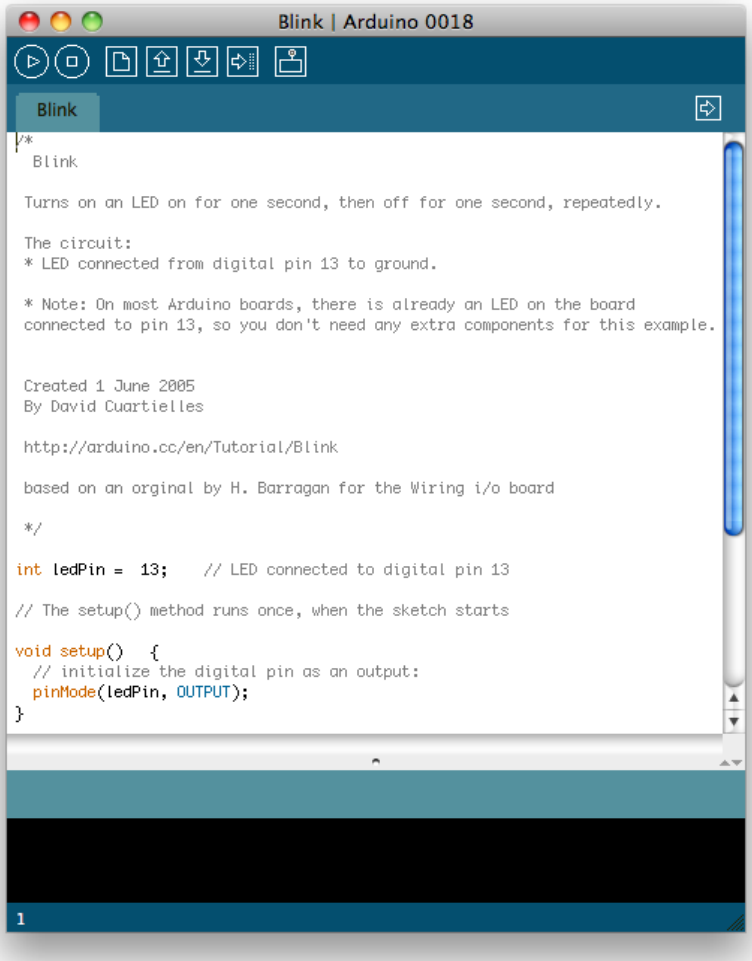


Figura 2.3.1 Interfaz del compilador de Arduino

Arduino utiliza para escribir el software lo que denomina "sketch" (*programa*). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/remplazar texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie:

Verify/Compile
Chequea el código en busca de errores.

Stop
Finaliza la monitorización serie y oculta otros botones

New
Crea un nuevo *sketch*.

Open
Presenta un menú de todos los programas *sketch* de su "sketchbook", (*librería de sketch*). Un click sobre uno de ellos lo abrirá en la ventana actual.

Nota: Debido a un error *bug* en Java, la barra de desplazamiento *scroll* no funciona en este menú; si necesita abrir un programa que no se vea en la lista, utilice File | Sketchbook en el menú.

Save
Salva el programa *sketch*.

Upload to I/O Board
Compila el código y lo vuelca en la placa E/S de Arduino. Ver volcado más detalles abajo.

Serial Monitor
Inicia la monitorización serie Monitorización Serie.

Encontrará otros comandos en los cinco menús: File, Edit, Sketch, Tools, Help. Los menús son sensibles al contexto, lo que significa que estarán disponibles sólo los elementos relevantes para la tarea que esté realizando en ese momento:

-Edit

Copy for Discourse
Copia el código de su *sketch* en el portapapeles para con el formato adecuado para publicarlo en un foro, incluyendo la sintaxis coloreada.

Copy as HTML
Copia el código de un programa (*sketch*) al portapapeles en formato HTML, adecuándolo para incrustarlo en una página web.

-Sketch

Verify/Compile
Verifica los errores de su programa (*sketch*).

Import Library
Añade una librería a su programa (*sketch*) insertando la sentencia #include en el código.

Show Sketch Folder
Abre la carpeta de programas (*sketch*) en el escritorio.

Add File...
Añade un fichero fuente al programa (se incluirá desde su ubicación actual). El fichero aparece en una nueva pestaña en la ventana del programa. Los ficheros pueden ser quitados del programa (*sketch*) utilizando el menú "tab".

-Tools

Auto Format
Da formato al código proporcionando estética: por ejemplo realiza tabulaciones entre la apertura y cierre de llaves, y las sentencias que tengan que ser tabuladas lo estarán.

Board
Selecciona la placa que estás usando. Ver más abajo en Descripción de las placas.

Serial Port
Este menú contiene todos los dispositivos de serie (reales o virtuales) de su equipo. Se refrescará automáticamente cada vez que abras el menú tools.

Burn Bootloader
Este elemento del menú le permite grabar un gestor de arranque (*bootloader*) dentro del micro-controlador de la placa Arduino. Aunque no es un requisito para el normal funcionamiento de la placa Arduino, le será útil si compra un nuevoATmega (el cual viene normalmente sin gestor de arranque). Asegúrese que ha seleccionado la placa correcta en el menú Boards antes de grabar el *bootloader*. Cuando use AVR ISP, tendrá que seleccionar en el menú Serial Port el puerto correspondiente.

-Sketchbook (Librería de Sketch)

El entorno de Arduino incluye el concepto de "sketchbook": que es el lugar estándar para el almacenamiento de sus programas (o "sketch"). Los "sketches" dentro de su "sketchbook" pueden abrirse desde el menú File > Sketchbook o desde el botón de la barra de herramientas Open. La primera vez que arranque el software Arduino, se creará un directorio para su "sketchbook". Puede visualizar o cambiar su localización dentro de "sketchbook location" desde el apartado Preferences.

-Tabs, Multiple Files, and Compilation (Pestañas, Ficheros múltiples y compilación)

Permite manejar "sketches" con más de un fichero (cada uno de los cuales aparece en su pestaña). Pueden ser normalmente ficheros de código Arduino (no extensiones), ficheros C (extensiones .c), ficheros c++ (.cpp), o ficheros de cabecera (.h).

-Uploading (Volcado)

Antes de volcar su "sketch", necesitará seleccionar los elementos correspondientes desde los menús Tools > Board yTools > Serial Port. Las boards (placas) están descritas abajo.

En los Mac, el puerto serie será probablemente algo como/dev/tty.usbserial-1B1 (para una placa USB), o /dev/tty.USA19QW1b1P1.1 (para una placa serie conectada con un adaptador Keyspan USB-to-Serial).

En Windows, probablemente sea COM1 o COM2 (para una placa serie) o COM4, COM5, COM7, o superior (para una placa USB)- para encontrarlos, debes buscar los dispositivos serie USB en la sección de puertos del Administrador de Dispositivos de Windows.

En Linux, debería ser /dev/ttyUSB0, /dev/ttyUSB1 o similar.

Una vez que ha seleccionado el puerto serie y la placa, presione el botón de volcado en la barra de herramientas o seleccione Upload to I/O Board desde el menú File.

Las actuales placas de Arduino se resetearán automáticamente y comenzará el volcado. Como las placas antiguas carecen de auto-reset, necesitará presionar el botón de reset en la placa, justo antes de iniciar el volcado. En muchas placas verá el led RX y TX parpadeando cuando el "sketch" está actualizándose. El entorno de Arduino mostrará un mensaje cuando el volcado esté completado, o mostrará un error.

Cuando se vuelca un "sketch", está utilizando el "bootloader" de Arduino, un pequeño programa que ha sido cargado en el micro-controlador en su placa. Permite el volcado del código sin utilizar hardware adicional. El "bootloader" está activo durante unos segundos cuando la placa es reseteada; después se inicia el "sketch" que más recientemente se hubiera actualizado en el micro-controlador. El "bootloader" produce un parpadeo en el LED de la placa (pin 13) cuando se inicia (p.e. cuando las placas son reseteadas)

-Third-Party Hardware (Hardware de terceros)

Se puede agregar soporte para hardware de terceros en el directorio hardware del directorio "sketchbook". Las plataformas instaladas aquí pueden incluir la definición de las placas (que aparecen en el menú board), librerías del núcleo, "bootloaders", y definiciones de programador. Para instalarla, cree un directorio hardware, y en él descomprima la plataforma de terceros con su directorio. (No utilices "arduino" como nombre del subdirectorio o sobre-escribirás la plataforma Arduino).

Para desinstalarlo, simplemente borre ese directorio. Y más detalles sobre la creación de paquetes de hardware de terceros visita Páginas de Plataformas en la web de Google Code developers.

-Serial Monitor (Monitor Serie)

Muestra los datos enviados desde la placa Arduino (placa USB o serie). Para enviar datos a la placa, teclee el texto y pulsa el botón "send" o enter. Seleccione la velocidad (baud rate) en el menú desplegable que coincida con el configurado enSerial.begin dentro de su "sketch".

Advertir que en Mac o Linux, la placa Arduino se resetea (su "sketch" es reiniciado desde el principio) cuando conecta con el monitor serie.

Se puede también comunicar con la placa desde Processing, Flash, MaxMSP, etc (ver Web de interface ('interfacing page') para más detalles)

-Preferences (Preferencias)

Pueden configurarse otras preferencias en el apartado preference (lo podrás encontrar bajo el menú Arduino para los Mac, o en File para Windows y Linux). El resto de opciones puede ser localizado en el fichero de preferencias, que se podrá encontrar dentro del mismo apartado preference.

-Boards (Placas)

La selección de placa tiene dos efectos: los parámetros utilizados cuando compila (por ejemplo, CPU usada y velocidad (baud rate)) y vuelcan los "sketches"; y el fichero y configuración utilizados por el gestor de arranque (*bootloader*)) al ser cargado. Algunas de las definiciones de las placas difieren sólo en lo segundo, incluso si ha sido cargado satisfactoriamente con una particular selección, usted podrá comprobarlo antes de grabar el "bootloader".

2.4 PROGRAMACIÓN Y FUNCIONES ESPECÍFICAS

Una de las desventajas que posee Arduino con respecto a un autómata convencional radica en su programación. Arduino se programa en el lenguaje C++. Por ello resulta más complicado, pero a su vez es mucho más versátil. Con Arduino se puede programar casi cualquier funcionamiento que se desee. Un ejemplo de ello es que podemos realizar la linealización de un sensor sin necesidad de circuitería externa, adquirir los datos del sensor y registrarlos en una memoria previamente asignándole un valor digital dentro de un rango definido en el convertidor analógico-digital del microcontrolador.

Los diferentes ejemplos de programación desarrollados para el control de semáforos vienen recogidos por orden de complejidad en el **Anexo B**.

Arduino admite la mayor parte de librerías usadas comúnmente en C++ y todas sus estructuras básicas, pero además la tarjeta de control nos proporciona funciones específicas para Arduino que son de gran ayuda al programador. A continuación se comenta la estructura de programa y las funciones específicas más relevantes en nuestro caso.

-setup () (Configuración)

La función setup () se llama cuando se inicia un programa. Se usa para inicializar las variables, los modos de contactos, comenzar a usar las bibliotecas, etc. La función de configuración sólo se ejecutará una vez, después de cada momento del encendido o reinicio de la placa Arduino.

-loop () (Bucle)

Después de crear una función setup (), que inicializa y establece los valores iniciales, el bucle () hace precisamente lo que sugiere su nombre, y los bucles de forma consecutiva, permitiendo que su programa para cambiar y responder. Se usa para controlar activamente la placa Arduino.

-pinMode()

Configura un pin como entrada o salida. Para utilizarla, le pasas el número del pin que vas a configurar y la constante INPUT u OUTPUT. Es decir configura el pin especificado para comportarse como una entrada o una salida. Se usa dentro de la función setup(). Sintaxis ->pinMode(pin, modo)

-digitalWrite()

Escribe o envía un valor HIGH o LOW hacia un pin digital. Por ejemplo, la línea: digitalWrite(ledPin, HIGH);

-digitalRead()

Lee el valor de un pin digital especificado, HIGH o LOW.
Sintaxis-> digitalRead(pin)
Parámetros->pin: el número de pin digital que quieres leer (*int*)
Devuelve HIGH o LOW.

-delay()

Pausa el programa por un tiempo determinado (en milisegundos) especificado por un parámetro. Hace a Arduino esperar por el número especificado de milisegundos antes de continuar con la siguiente línea. Hay 1000 milisegundos en un segundo, por lo que la línea siguiente crea un retraso de un segundo: delay(1000).

- attachInterrupt(interrupcion, funcion, modo)

Especifica la función a la que invocar cuando se produce una interrupción externa. Reemplaza cualquier función previa que estuviera enlazada a la interrupción. La mayoría de las placas Arduino tienen dos interrupciones externas: Las número 0 (en el pin digital 2) y la 1 (en el pin digital 3).

Parámetros:

- interrupción: el número de la interrupción (*int*)
- función: la función a la que invocar cuando la interrupción tiene lugar; esta función no debe tener parámetros ni devolver nada. Esta función es a veces referenciada como *rutina de interrupción de servicio*
- modo define cuando la interrupción debe ser disparada. Hay cuatro constantes predefinidas como valores válidos:

LOW para disparar la interrupción en cualquier momento que el pin se encuentre a valor bajo (LOW).
CHANGE para disparar la interrupción en cualquier momento que el pin cambie de valor.
RISING para disparar la interrupción cuando el pin pase de valor bajo (LOW) a alto (HIGH).
FALLING para cuando el pin cambie de valor alto (HIGH) a bajo (LOW)

-detachInterrupt(interrupt)

Apaga la interrupción dada. Interrupt: el número de interrupción a invalidar (0 o 1).

-noInterrupts()

Desactiva las interrupciones (pueden reactivarse usando interrupts()). Las interrupciones permiten que las operaciones importantes se realicen de forma transparente y están activadas por defecto. Algunas funciones no funcionarán y los datos que se reciban serán ignorados mientras que las interrupciones estén desactivadas. Las interrupciones pueden perturbar ligeramente el tiempo de temporizado, sin embargo puede que sea necesario desactivarlas para alguna parte crítica del código.

- interrupts()

Activa las interrupciones (después de haberlas desactivado con noInterrupts(). Las interrupciones permiten que se ejecuten ciertas tareas en segundo plano que están activadas por defecto. Algunas funciones no funcionarán correctamente mientras las interrupciones estén desactivadas y la comunicación entrante puede ser ignorada. Las interrupciones pueden perturbar ligeramente la temporización en el código y deben ser desactivadas sólo para partes particularmente críticas del código.

2.5 CARACTERÍSTICAS TÉCNICAS DE E/S

- **Propiedades de los Pines Configurados como Entrada**

Los pines de Arduino (Atmega) por defecto son de entrada, por lo que es necesario configuraros explícitamente como entradas con pinMode(). Se dice que los pines configurados como entradas están en estado de alta impedancia. Una forma de explicar esto es que los terminales de entrada hacen demandas extremadamente pequeñas en el circuito que están muestreando, se dice que equivale a una resistencia en serie de 100 megaohmio frente al pin. Esto significa que se necesita muy poca corriente para pasar el pin de entrada de un estado a otro, y puede hacer posible el uso de los pines para tareas como la utilización de un sensor capacitivo al tacto, la lectura de un LED como un fotodiodo, o la lectura de un sensor analógico con un esquema como el RCTime.

Esto también significa sin embargo, que los terminales de entrada sin conectar nada a ellos, o con los cables conectados a ellos sin estar conectados a otros circuitos, reflejarán cambios aparentemente aleatorios en el estado de pin, recogiendo el ruido eléctrico del entorno, o el acoplamiento capacitivo del estado de un pin próximo.

-Resistencias Pullup

A menudo es útil para colocar un pin de entrada en un estado conocido si no hay un estado de entrada. Puede hacerse añadiendo una resistencia pull-up (a +5 V), o una resistencia pull-down (resistencia a tierra) en la entrada, 10K suele ser un valor muy común.

También hay resistencias pullup de 20K conveniente integradas en el chip Atmega a las que se puede acceder desde el software. Estos resistencias pull-up incorporadas son accedidas de la siguiente manera.

```
pinMode(pin, INPUT);          // pone el pin como entrada
digitalWrite(pin, HIGH);      // activa la resistencia pullup
```

Ten en cuenta que las resistencias pull-up proporcionan suficiente corriente para dar una luz tenue con un LED conectado a un pin que se ha configurado como entrada. Si el LED de un proyecto parece estar funcionando pero muy tenuemente, es posible que sea esto lo que está pasando, y el programador ha olvidado usar pinMode() para ajustar los pines como salidas.

También debes tener en cuenta que las resistencias pull-up son controladas por los mismos registros (posiciones de memoria interna del chip) que controlan si un pin está alto (HIGH) o bajo (LOW). Por consiguiente, un pin que se configura para tener las resistencias pullup activadas cuando está configurado como entrada, debe tener el pin a alto (HIGH) si el pin es cambiado como salida (OUTPUT) con pinMode(). Esto funciona en la otra dirección también, y un pin de salida que queda en un estado alto tendrá las resistencias pull-up activas, si cambia a entrada (INPUT) con pinMode().

NOTA: El pin Digital 13 es más difícil de usar que otros pines digitales porque tiene un LED y una resistencia asociada soldados a la placa en la mayoría de las placas. Si activa la resistencia pull-up 20k del interior, se pondra en alrededor de 1,7 V en lugar de los 5V que se esperan debido a que el LED integrado y la resistencia en serie bajan el nivel del voltaje, lo que se traduce en que siempre retornará bajo (LOW). Si estás obligado a usar el pin 13 como entrada digital, utiliza una resistencia pulldown externa.

▪ Propiedades de los Pines Configurados como salida

Los pines configurados como salida *(OUTPUT)* con pinMode() se dice que están en un estado de baja impedancia. Esto significa que puede proporcionar una cantidad sustancial de corriente a otros circuitos. Los pines del Atmega pueden proporcionar corriente positiva o proporcionar corriente negativa de hasta 40 mA (miliamperios) a otros dispositivos o circuitos. Esta es suficiente corriente para la brillante luz de un LED (no te olvides de la resistencia en serie), o para utilizar muchos sensores por ejemplo, pero no es corriente suficiente para utilizar la mayoría de relés, solenoides o motores.

Los cortocircuitos en los pines de Arduino, o intentos de extraer mucha corriente de ellos, pueden dañar o destruir los transistores de salida en el pin, o dañar completamente el chip Atmega. A menudo, esto se traducirá en un pin del micro-controlador "muerto", pero el resto del chip seguirá funcionando adecuadamente. Por esta razón es buena idea conectar los pines de salida a otros dispositivos con resistencias de 470Ω o 1k, limitando la corriente máxima que desde los pines es requerida para una aplicación particular.

-RCtime

En las situaciones en las que se usan todos los pines A/D (analógicos/digitales) de las *Duinos*, RCtime es una solución para leer cualquier tipo de sensores resistivos en cualquier pin digital.

Esta función RCtime duplica la función de Basic Stamp del mismo nombre. Se puede utilizar para leer los sensores resistivos de cualquier tipo. Cambia el tamaño del condensador para lograr la resolución deseada.

La función también puede utilizarse para leer el voltaje de salida de los sensores, tales como el sensor de distancia por infrarrojos Sharp con algunas advertencias.

Una virtud de RCtime es que puede tener un rango muy amplio, y manejar valores que para ser leidos requieren una entrada A/D de 16-18 bits. Una desventaja es que no es perfectamente lineal, porque la carga de un condensador a través de una resistencia no produce una curva lineal.

```
/* RCtime
 *   Duplicates the functionality of the Basic Stamp's RCtime
 *   Allows digital pins to be used to read resistive analog sensors
 *    One advantage of this technique is that is can be used to read
very wide ranging inputs.
 *   (The equivalent of 16 or 18 bit A/D)
 *
```

```
    Schematic
                      +5V
                       |
                       |
                      ___
                      ___     Sensing Cap
                       |          .001 ufd   (change to suit for required
resolution)
                       |       (102) pfd
                       |
  sPin ---\/\/\/-----.
         220 - 1K     |
                      |
                      \
                      /      Variable Resistive Sensor
                      \      Photocell, phototransistor, FSR etc.
                      /
                      |
                      |
                      |
                    _____
                     ___
                      _

  */
  int sensorPin = 4;                // 220 or 1k resistor connected to
this pin
  long result = 0;
  void setup()                     // run once, when the sketch starts
  {
     Serial.begin(9600);
     Serial.println("start");      // a personal quirk
  }
  void loop()                      // run over and over again
  {

     Serial.println( RCtime(sensorPin) );
     delay(10);
  }
```

```
  long RCtime(int sensPin){
     long result = 0;
     pinMode(sensPin, OUTPUT);       // make pin OUTPUT
     digitalWrite(sensPin, HIGH);      // make pin HIGH to discharge
capacitor - study the schematic
     delay(1);                         // wait a  ms to make sure cap is
discharged

     pinMode(sensPin, INPUT);         // turn pin into an input and time
till pin goes low
     digitalWrite(sensPin, LOW);      // turn pullups off - or it won't
work
     while(digitalRead(sensPin)){     // wait for pin to go low
        result++;
     }

     return result;                    // report results
  }
```

2.6 DIFERENTES VERSIONES DE PLACAS ARDUINO

Hay multitud de diferentes versiones de placas Arduino. La actual placa básica, el Duemilanove, usa Atmel ATmega328. La anterior Diecimila, y las primeras unidades de Duemilanove usaban el Atmel ATmega168, mientras que las placas más antiguas usan el ATmega8. El Arduino Mega está basado en el ATmega1280.

A continuación veremos las diferentes versiones de placas Arduino junto con sus características básicas.

Arduino Duemilanove

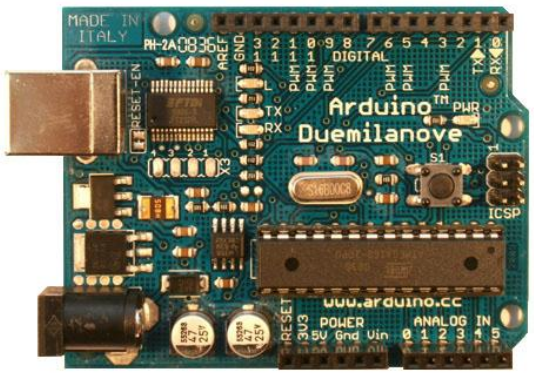


Figura 2.6.1 Arduino Duemilanove

Visión general

El Arduino Duemilanove ("2009") es una placa con microcontrolador basada en el ATmega168 o elATmega328. Tiene 14 pines con entradas/salidas digitales (6 de las cuales pueden ser usadas como salidas PWM), 6 entradas analógicas, un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación, una cabecera ISCP, y un botón de reset. Contiene todo lo necesario para utilizar el microcontrolador; simplemente conéctalo a tu ordenador a través del cable USB o aliméntalo con un transformador o una batería para empezar a trabajar con él.

Características

Microcontrolador	ATmega368 (ATmega168 en versiones anteriores)
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines E/S digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Intensidad por pin	40 mA
Intensidad en pin 3.3V	50 mA
Memoria Flash	16 KB (ATmega168) o 32 KB (ATmega328) de las cuales 2 KB las usa el gestor de arranque(bootloader)
SRAM	1 KB (ATmega168) o 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) o 1 KB (ATmega328)
Velocidad de reloj	16 MHz

Arduino Diecimila

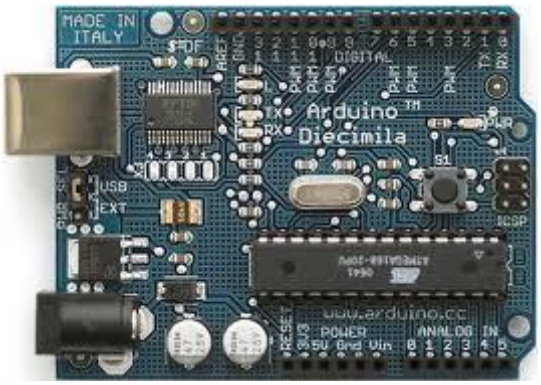


Figura 2.6.2 Arduino Diecimila

Visión general.

La Arduino Diecimila es una placa microcontroladora basada en el chip ATmega168. Tiene 14 E/S digitales (6 de las cuales se puedes utilizar como salidas PWM), 6 entradas analógicas, un cristal de 16MHz, conexión USB y botón de reseteo. Contiene todo lo necesario para el soporte del microcontrolador; simplemente conectala a un ordenador con un cable USB o alimentala con un adaptador AC/DC o una batería y comenzará a funcionar.

Características.

Microcontrolador	ATmega168
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (limites)	6-20 V
Pines E/S Digitales	14 (de ellos 6 son salidas PWM)
Pines de entrada Analógica	6
Intensidad por pin de E/S	40 mA
Intensidad por pin de 3.3V	50 mA
Memoria Flash	16 KB (2 KB reservados para el gestor de arranque)
SRAM	1 KB
EEPROM	512 bytes
Velocidad del reloj	16 MHz

Arduino Nano

Figura 2.6.3 Arduino Nano

Descripción General

El Arduino Nano es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o ATmega168 (Arduino Nano 2.x) que se usa conectándola a una protoboard. Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B en vez del cable estándar. El nano fue diseñado y está siendo producido por Gravitech.

Características

Microcontrolador	Atmel ATmega168 o ATmega328
Tensión de Operación (nivel lógico)	5 V
Tensión de Entrada (recomendado)	7-12 V
Tensión de Entrada (límites)	6-20 V
Pines E/S Digitales	14 (de los cuales 6 proveen de salida PWM
Entradas Analógicas	8
Corriente máx por cada PIN de E/S	40 mA
Memoria Flash	16 KB (ATmega168) o 32 KB (ATmega328) de los cuales 2KB son usados por el bootloader
SRAM	1 KB (ATmega168) o 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) o 1 KB (ATmega328)
Frecuencia de reloj	16 MHz
Dimensiones	18,5mm x 43.2mm

Arduino Mega

Figura 2.6.4 Arduino Mega

Vision General

El Arduino Mega es una placa microcontrolador basada ATmeg1280. Tiene 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset. Contiene todo lo necesario para hacer funcionar el microcontrolador; simplemente conéctalo al ordenador con el cable USB o aliméntalo con un trasformador o batería para empezar. El Mega es compatible con la mayoría de shields diseñados para el Arduino Duemilanove o Diecimila.

Características

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

Arduino BT

Figura 2.6.5 Arduino bluetooth

Visión general

El Arduino BT es una placa Arduino con el módulo Blueetooh incorporado, que permite la comunicación inalámbrica. El módulo Bluetooth utilizado es el Bluegiga WT11, la versión iWrap.

El módulo Bluetooth se puede configurar con comandos enviados a través del puerto serie del ATmega168 (consulta la guía del usuario iWrap para más detalles). Un programa para configurar el nombre y código del módulo bluetooth se ejecuta una vez en cada BT Arduino. El nombre se establece en ARDUINOBT y el código de acceso en 12345. El ATmega168 viene precargado con un gestor de arranque que te permite subir *sketches* al consejo de administración a través de bluetooth. El código fuente del gestor de arranque está disponible en el repositorio SVN de Arduino.

LilyPad Arduino

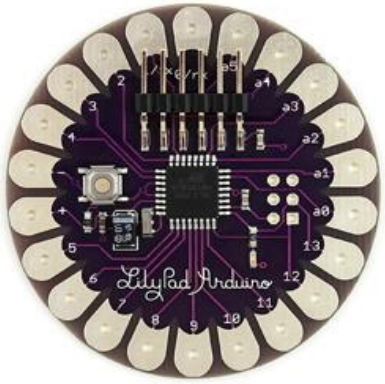


Figura 2.6.6 Arduino LilyPad

Visión general

El LilyPad Arduino es una placa con microcontrolador diseñado para prendas de ropa. Puede utilizar con complementos similares como fuentes de alimentación, sensores actuadores unidos por hilo conductor. La placa está basada en el ARmega168V (la versión de baja consumo del ATmega168), o el ATmega328V. ElLilyPad Arduino ha sido diseñado y desarrollado por Leah Buechley y SparkFun Electronics.

Características

Atención: *No alimentes el LilyPad Arduino con más de 5,5 voltios, ni inviertas la polaridad al conectarlo: Lo matarás.*

Microcontrolador	ATmega168V o ATmega328V
Voltaje de funcionamiento	2.7-5.5 V
Voltaje de entrada	2.7-5.5 V
Pines E/S Digitales	14 (de las cuales 6 proporcionan salida PWM)
Pines Entradas Analógicas Input Pins	6
Intensidad por pin	40 mA
Memora Flash	16 KB (de las cuales 2 KB las usa el gestor de arranque(bootloader))
SRAM	1 KB
EEPROM	512 bytes
Velocidad de reloj	8 MHz

Arduino Fio

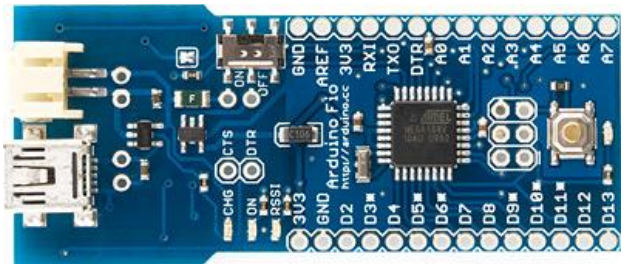


Figura 2.6.7 Arduino Fio

Información General

El Arduino Fio es una placa para microcontrolador basada en el ATmega328P. Funciona a 3.3V y 8MHz. Tiene 14 pines de E/S digitales (de los cuales 6 pueden usarse como salidas PWM), 8 entradas analógicas, un *resonador* en placa, un botón de reinicio (*reset*), y agujeros para montar conectores de pines. Tiene conexiones para una batería de polímero de Litio e incluye un circuito de carga a través de USB. En el reverso de la placa tiene disponible un zócalo para módulos XBee.

El Arduino FIO está diseñado para aplicaciones inalámbricas. El usuario puede subir sus *sketches* con un cable FTDI o una placa adicional adaptadora Sparkfun. Además, si utiliza un adaptador de USB a XBee modificado, como el USB Explorador de XBee, el usuario puede subir *sketches* de forma inalámbrica. La tarjeta viene sin conectores pre-montados, permitiendo el uso de diversos tipos de conectores o la soldadura directa de los cables.

Características

Microcontrolador	ATmega328P
Voltaje de trabajo	3.3V
Voltaje de Entrada	3.35 -12 V
Voltaje de Entrada en Carga	3.7 - 7 V
Pines E/S Digital	14 (of which 6 provide PWM output)
Pines de Entrada Analógica	8
Corriente DC por pin E/S	40 mA
Memoria Flash	32 KB (of which 2 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de Reloj	8 MHz

Arduino Mini

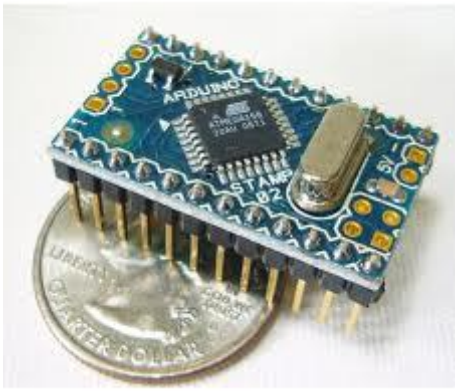


Figura 2.6.8 Arduino Mini

Información General

Arduino Mini es una placa con un pequeño microcontrolador basada en el ATmega168, pensada para ser usada en placas de prototipado y donde el espacio es un bien escaso. Puede ser programada con el adaptador Mini USB u otros adaptadores USB o RS232 a TTL serial.

Advertencia: No Alimente el Arduino mini con más de 9 voltios, o conecte la alimentación al revés.

Características

Microcontrolador	ATmega168
Voltaje de funcionamiento	5V
Voltaje de entrada	7-9 V
Pines E/S digital	14 (de las cuales 6 pueden ser usadas como salidas PWM)
Pines entrada analógica	8 (de las cuales 4 se extienden en pines)
DC Corriente continua por pin E/S	40 mA
Memoria Flash	16 KB (de las cuales 2 KB son usadas por el bootloader)
SRAM	1 KB
EEPROM	512 bytes
Velocidad de reloj	16 MHz

Arduino Pro

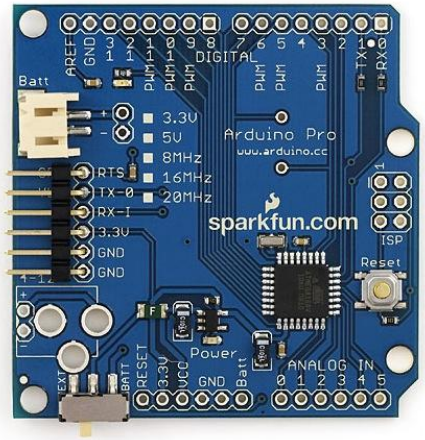


Figura 2.6.9 Arduino Pro

Información principal.

La Arduino pro es una placa con un microcontrolador ATmega168 o el ATmega328. La Pro viene en versiones de 3.3v / 8 MHz y 5v / 16 MHz. Tiene 14 E/S digitales (6 de las cuales se puedes utilizar como salidas PWM), 6 entradas analógicas, un resonador interno, botón de reseteo y agujeros para el montaje de tiras de pines. Vienen equipada con 6 pines para la conexión a un cable FTDI o a una placa adaptadora de la casa Sparkfun para dotarla de comunicación USB y alimentación.

La Arduino Mini Pro está destinada a instalaciones semi-permanentes en objetos o demostraciones. La placa viene sin conectores montados, permitiendo el uso de varios tipos de conectores o soldado directo de cables según las necesidades de cada proyecto en particular. La distribución de los pines es compatible con los shields de Arduino. Las versiones de 3.3v de la pro pueden ser alimentadas por baterías.

Características.

Microcontrolador	ATmega168 o ATmega328
Voltaje de entrada	3.35 -12v (en el modelo de 3.3v) o 5 - 12v (en el modelo de 5v)
Pines digitales de E/S	14 (6 de los cuales tienen salida PWM)
Intensidad máxima por E/S	40 mA
Memoria Flash	16KB en el ATmega168 y 32KB con el ATmega328 (de los cuales 2KB están reservados por el gestor de arranque)
SRAM	1KB
EEPROM	512 bytes
Velocidad de Reloj	8 MHz (modelo de 3.3v) o 16 MHz (modelo de 5v)

Arduino Serial

Figura 2.6.10 Arduino conexión serie

Descripción general

Es una placa básica que utiliza una interfaz RS232 para comunicarse con el ordenador o para la carga de sketches. Esta placa es fácil de montar, incluso como ejercicio de aprendizaje. Se ha diseñado para utilizar los componentes más simples posibles, de manera que sea fácil de construir, incluso si buscas las componentes en la tienda de la esquina.

Características.

Microcontrolador	ATmega8
Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (limites)	6-20 V
Pines E/S Digitales	14 (de ellos 6 son salidas PWM)
Pines de entrada Analógica	6
Intensidad por pin de E/S	40 mA
Intensidad por pin de 3.3V	50 mA
Velocidad del reloj	16 MHz

2.7 LOS SHIELDS DE ARDUINO

Los Shields son placas que se colocan encima de la placa Arduino y que amplían una nueva función para que sea controlada desde Arduino, para controlar diferentes aparatos, adquirir datos, etc.

Xbee Shield

La Xbee shield permite a una placa Arduino comunicarse de forma inalámbrica usando Zigbee. Está basada en el módulo Xbee de MaxStream. El módulo puede comunicarse hasta 100ft (30 metros) en interior o 300ft (90 metros) al aire libre (en visión directa). Puede ser usado como reemplazo del puerto serie/usb o puedes ponerlo en modo de comandos y configurarlo para una variedad de opciones de redes broadcast o malladas. La shield tiene pistas desde cada pin del Xbee hasta un orificio de soldar. También provee conectores hembra para usar los pines digitales desde 2 hasta 7 y las entradas analógicas, las cuales están cubiertas por la shield (los pines digitales de 8 a 13 no están cubiertos por la placa, así que puedes usar los conectores de la placa directamente). Podemos ver su aspecto en la figura 2.7.1.

Figura 2.7.1 Arduino Xbee Shield

Motor Control v1.1

Este shield permite a Arduino controlar motores eléctricos de corriente continua, servos y motores paso a paso y leer encoders. Esta shield está todavía bajo testeo, pero puedes echarle un ojo al esquema, y usarlo en tus diseños. Podemos ver su aspecto en la figura 2.7.2.

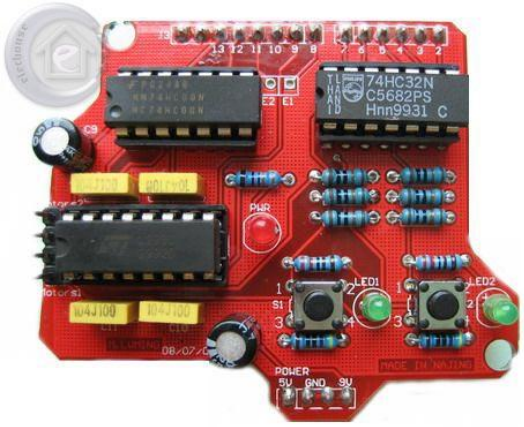


Figura 2.7.2 Arduino Motor Shield

Arduino Ethernet Shield

La Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Está basada en el chip ethernet Wiznet W5100. El Wiznet W5100 provee de una pila de red IP capaz de TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Usa la librería Ethernet para escribir programas que se conecten a internet usando la shield.

Por favor notar que el actual diseño de la ethernet shield no es compatible con el Arduino Mega.

La ethernet shield dispone de unos conectores que permiten conectar a su vez otras placas encima y apilarlas sobre la placa Arduino.

Arduino usa los pines digitales 10, 11, 12, y 13 (SPI) para comunicarse con el W5100 en la ethernet shield. Estos pines no pueden ser usados para e/s genéricas.

La shield provee un conector ethernet estándar RJ45

El botón de reset en la shield resetea ambos, el W5100 y la placa Arduino.

La shield contiene un número de LEDs para información:

- PWR: indica que la placa y la shield están alimentadas
- LINK: indica la presencia de un enlace de red y parpadea cuando la shield envía o recibe datos
- FULLD: indica que la conexión de red es full duplex
- 100M: indica la presencia de una conexión de red de 100 Mb/s (de forma opuesta a una de 10Mb/s)
- RX: parpadea cuando la shield recibe datos
- TX: parpadea cuando la shield envía datos
- COLL: parpadea cuando se detectan colisiones en la red

El jumper soldado marcado como "INT" puede ser conectado para permitir a la placa Arduino recibir notificaciones de eventos por interrupción desde el W5100, pero esto no está soportado por la librería Ethernet. El jumper conecta el pin INT del W5100 al pin digital 2 de Arduino.

El slot SD en la shield no está soportado por el software Arduino. En la figura 2.7.3 podemos ver el aspecto de esta placa.

Figura 2.7.3 Arduino Ethernet Shield.

Capítulo 3

Desarrollo del programa de control.

3.1 INTRODUCCIÓN

La finalidad de este proyecto, en cuanto a programación se refiere, consiste en diseñar un programa en Lenguaje C que ejemplifique la situación mostrada en la figura 3.1. Se trata de un cruce de tres semáforos de vehículos y uno de peatones. Además tenemos un pulsador para el semáforo peatonal y una barrera de infrarrojos para la detección de vehículos.

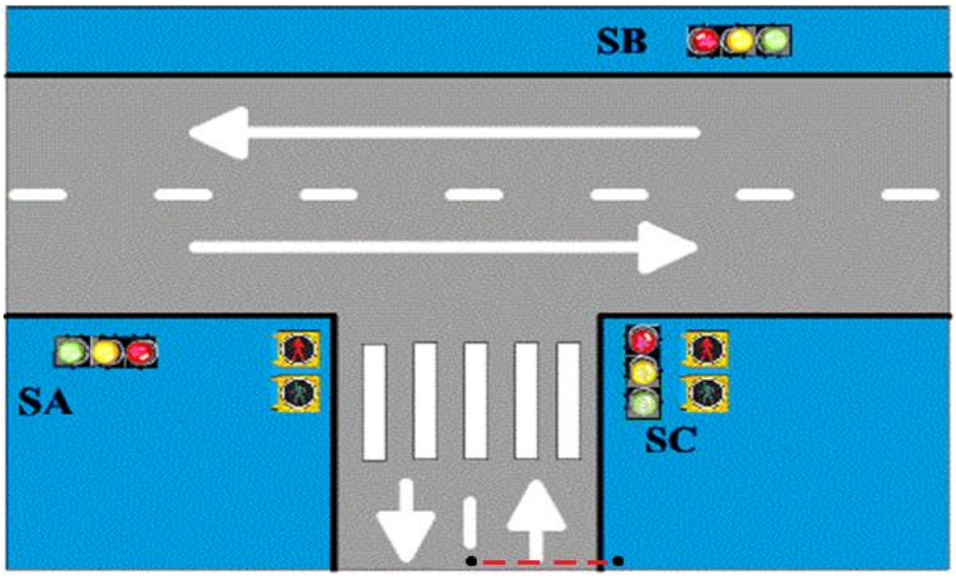


Figura 3.1 Cruce de controlado por semáforos

3.2 GRAFCET Y FLUJOGRAMAS

A continuación se muestra el Grafcet utilizado como punto de partida para el desarrollo del programa de control a cargar en Arduino. En primer lugar se enumeran las variables internas (tabla 3.2.2), las variables externas (tabla 3.2.1) y las acciones a llevar a cabo (tabla 3.2.3).

VARIAB. EXTERNAS	DESCRIPCIÓN
x	Pulsador peatonal
y	Detección barrera de infrarrojos

Tabla 3.2.1 Variables externas

VARIAB. INTERNAS	DESCRIPCIÓN
T1	Tiempo de espera para el verde del semáforo A
T2	Tiempo de espera para el amarillo del semáforo A
T3	Tiempo de espera para el rojo del semáforo A
T4	Tiempo de espera para el verde del semáforo B
T5	Tiempo de espera para el amarillo del semáforo B
T6	Tiempo de espera para el rojo del semáforo B
T7	Tiempo de espera para el verde del semáforo C
T8	Tiempo de espera para el amarillo del semáforo C
T9	Tiempo de espera para el rojo del semáforo C
Td	Tiempo de descuento del display
Tp	Tiempo de espera tras acabar el semáforo peatonal

Tabla 3.2.2 Variables internas

ACCIONES	DESCRIPCIÓN
VA	Semáforo A en verde
AA	Semáforo A en amarillo
RA	Semáforo A en Rojo
VB	Semáforo B en verde
AB	Semáforo B en amarillo
RB	Semáforo B en Rojo
VC	Semáforo C en verde
AC	Semáforo C en amarillo
RC	Semáforo C en Rojo
VP	Semáforo de peatones en verde
RP	Semáforo de peatones en rojo
Display On	Display encendido
Display Off	Display apagado
Reset x	Pone x a cero
Reset y	Pone y a cero

Tabla 3.2.2 Acciones

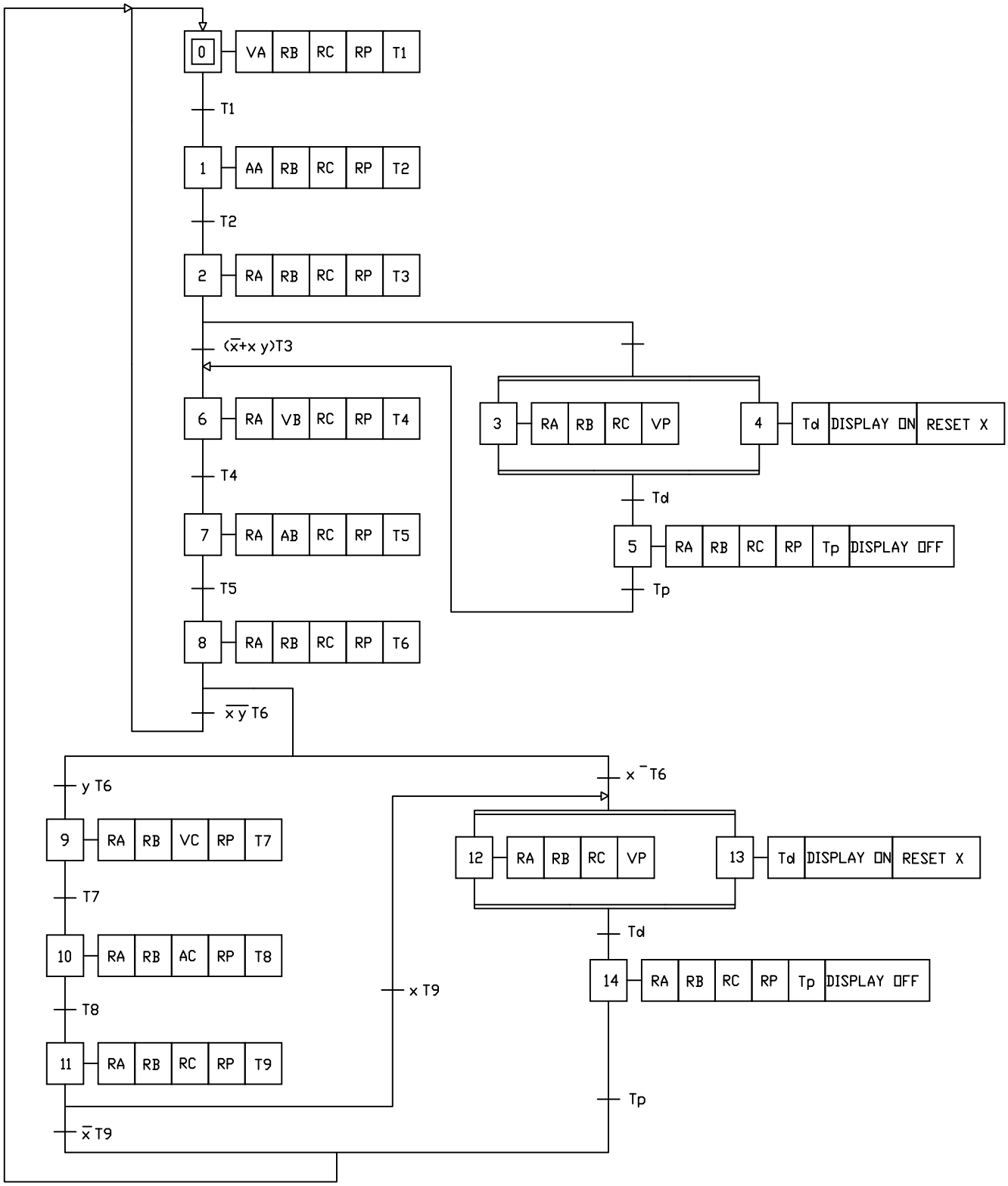


Figura 3.2.1 Grafcet

Tanto en el Grafcet de la figura 3.2.1 como en el código de programación las variables x e y se definen como:

"x" es la variable declarada como volátil inicializada a 0 la cual es modificada al valor de 1 por la función "pulse" llamada por la interrupción 0 (pin2).

"y" es la variable declarada como volátil inicializada a 0 la cual es modificada al valor de 1 por la función "sensor" llamada por la interrupción 1 (pin3).

Explicación del Grafcet:

Al iniciarse el programa (tras conectar Arduino a alimentación) se activa el semáforo A (etapas 0, 1 y 2). Al acabar este semáforo (rojo activado), se ejecuta la condición de que si x=1 (se ha pulsado el botón del semáforo de peatones) y no hay detección por el sensor de infrarrojos (y=0) se ejecutan a la vez las etapas 3 y 4 que corresponden al semáforo peatonal y al display de cuenta atrás respectivamente. En caso de ser x=0 entonces se activa el semáforo B (etapas 6,7 y 8).

Una vez este semáforo termine (rojo activado), se ejecuta una condición múltiple. Si "x" e "y" no están activos se vuelve a ejecutar el semáforo A, si x=1 y y=0 se activa el semáforo de peatones y el display, y si y=1 (hay vehículos por la calle de poco tránsito) se pondrá en verde el semáforo C (etapas 9, 10 y 11). Después en caso de estar activo "x" se ejecutarían las etapas 12 y 13, y después la 14 que al acabar vuelve a activar el semáforo A. Si después de acabar el semáforo C no está activa "x" directamente se vuelve a la etapa inicial.

Flujogramas

También se han empleado, por otro lado, flujogramas para el diseño del programa. Ha de constar que los tiempos de los temporizadores no son reales para economizar tiempo en la demostración del programa en el panel de pruebas o maqueta.

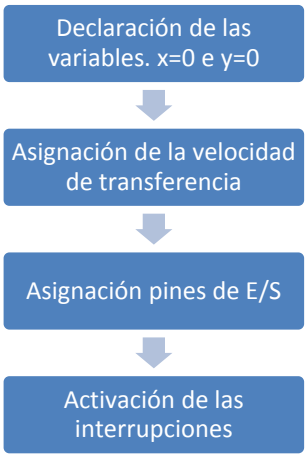


Figura 3.2.2 Establecimiento de condiciones iniciales

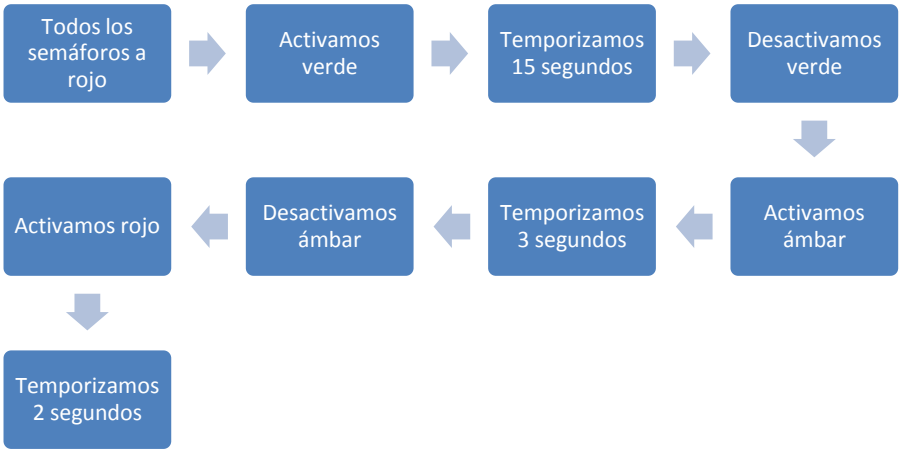


Figura 3.2.3 Semáforos de vehículos

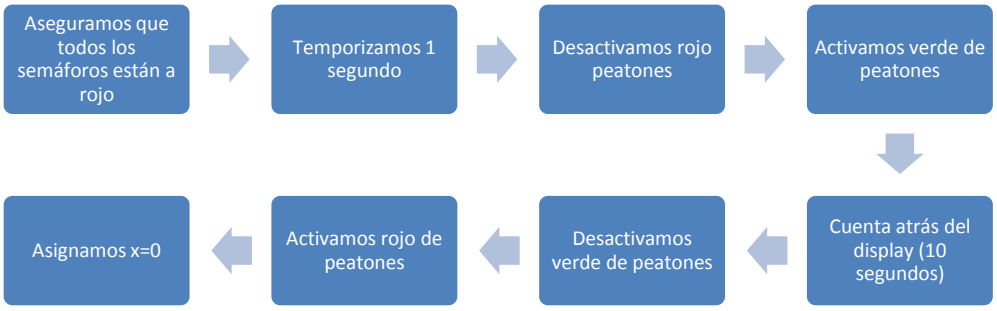


Figura 3.2.4 Semáforo peatonal

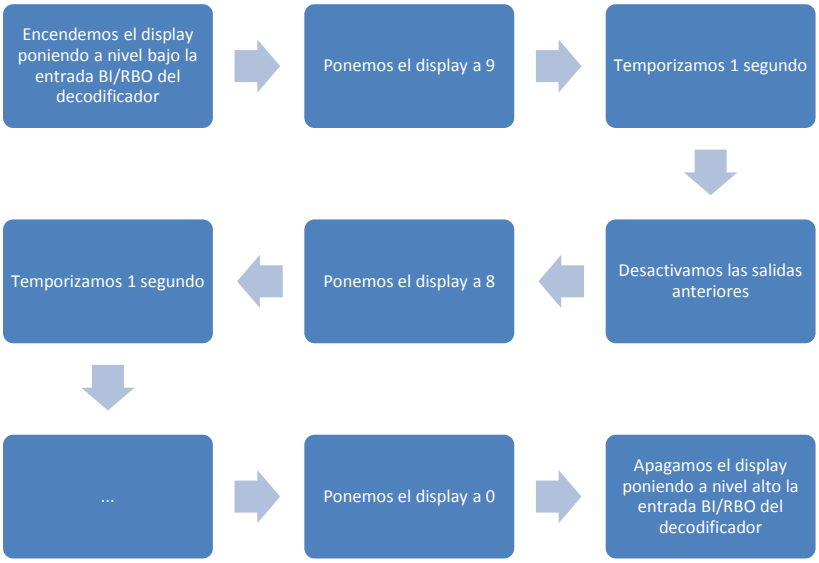


Figura 3.2.5 Display de cuenta atrás

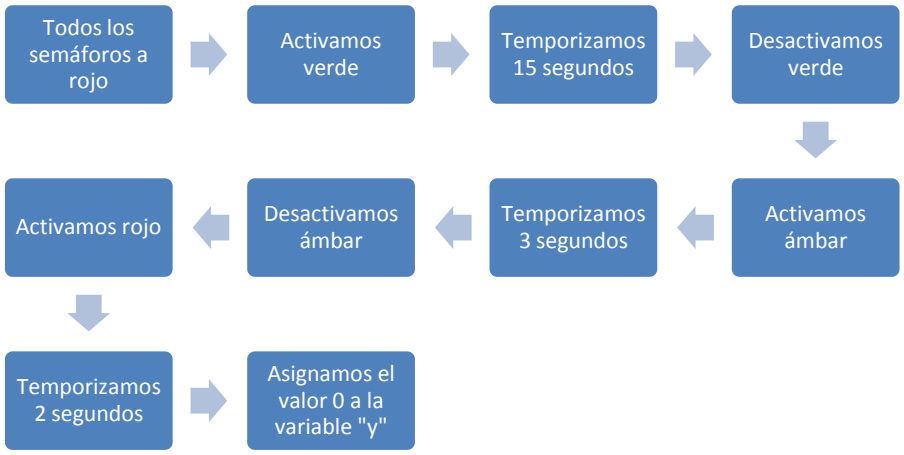


Figura 3.2.6 Semáforo de vehículos de la calle con poco tránsito (semáforo C)

3.3 PROBLEMAS Y SOLUCIONES ENCONTRADOS

En el momento de la programación surgieron dos problemas muy a tener en cuenta para este ejemplo como para cualquier otro programa a desarrollar.

Como ya es conocido la programación estructural se ejecuta en secuencia. Las sentencias que deseamos que se ejecuten han de estar dentro del cuerpo de la función loop(), pero en muchas ocasiones es conveniente e incluso obligatorio por las características del código diseñado, poner funciones fuera del bucle loop.

Pues bien, cuando llamas a una de estas funciones de fuera del bucle loop suele ocurrir que aparte de ejecutarse esa función a la que hemos llamado se siga ejecutando el código que haya a continuación. Este indeseado comportamiento ha de resolverse con un correcto orden del código de programación e incluso usando estructuras que nos interrumpan la ejecución secuencial del programa.

Un ejemplo de estas estructuras es la de salto incondicional "GO TO", la cual se recomienda intentar no usar a menos que no tengamos otra opción.

Por otro lado, ocurre que mientras se está ejecutando una temporización, hasta que esta no termine no podemos leer una entrada usando digitalRead() o analogRead(). Como tanto el pulsador de peatones como la barrera de infrarrojos no se mantienen enclavados sino que ofrecen un pulso de frecuencia indefinida (normalmente breve), dando igual que parte de nuestro programa se esté leyendo en ese instante, se hace necesario el uso de interrupciones.

Debido a que no podemos usar la función delay() dentro del argumento de la función que definimos en la sintaxis de la interrupción, se intentó evitar su uso.

Las dos ideas más relevantes para enclavar nuestras entradas hasta que se ejecutase una estructura condicional y así poder leer los pines de entrada son:

- Uso de un **biestable tipo D** para enclavar un valor de tensión. El mayor problema, su dependencia del estado anterior de la salida. Solución fallida.

- Enclavar el nivel de tensión usando la carga y descarga de **un condensador**. Necesitaríamos un condensador o grupo de condensadores de capacidades del orden de 50mF y ajustar los valores de tensión. Además si se quisiera cambiar el tiempo de funcionamiento de los semáforos habría que rediseñar dicha red. Solución fallida. Se muestra a continuación el circuito (figura 3.3.1) y su simulación (figura 3.3.2).

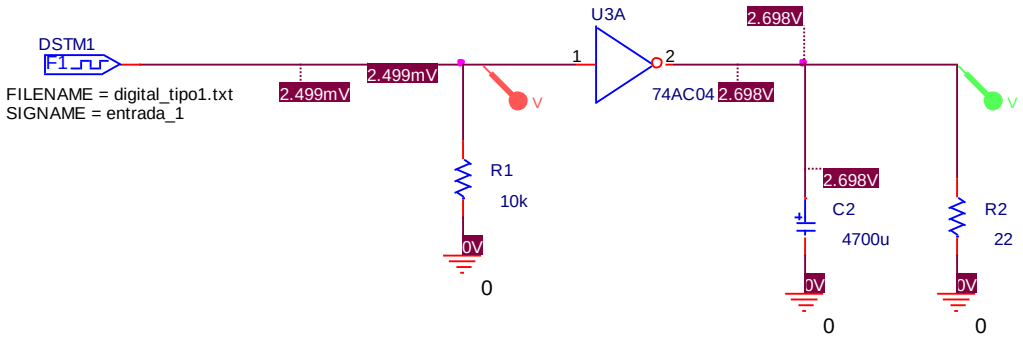


Figura 3.3.1 Circuito de enclavamiento por condensador

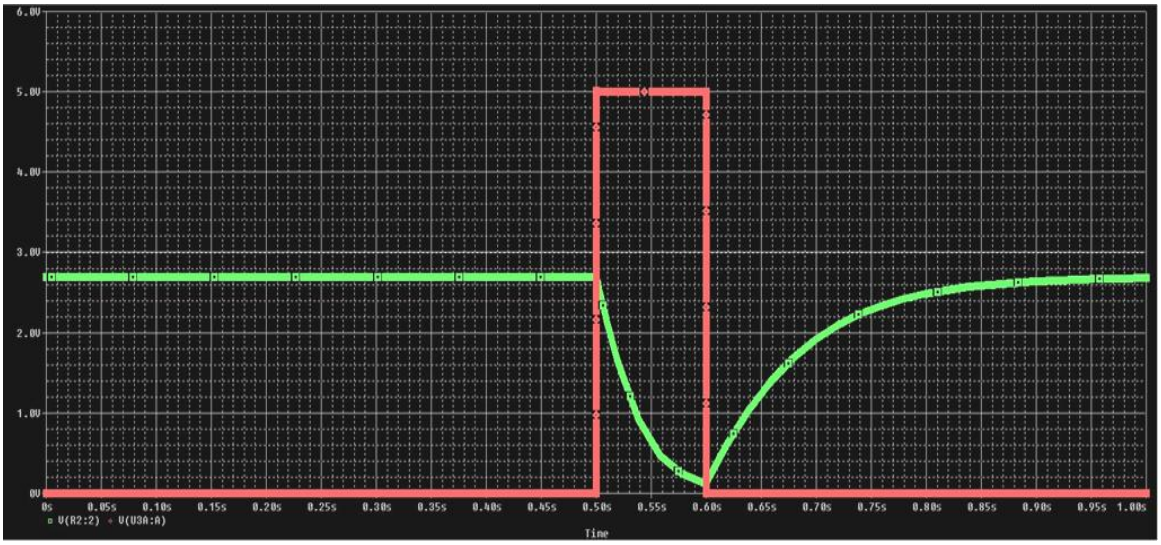


Figura 3.3.2 Simulación circuito de la figura 3.3.1

La línea verde corresponde a la salida del circuito y la roja al pulso de entrada. Como se puede apreciar se ha conseguido enclavar para nivel bajo de tecnología TTL la respuesta de salida a el doble de la duración del pulso con un condensador de 4700uF. Lo cual es insuficiente en una situación real.

Por todo ello y para evitar coste adicional la solución perfecta es programar interrupciones. Arduino sólo consta de dos interrupciones posibles. Esta es una desventaja pues en otros sistemas necesitaríamos un microcontrolador con más interrupciones o utilizar varios de ellos.

Capítulo 4

El panel de pruebas.

4.1 INTRODUCCIÓN

Uno de los principales objetivos de este proyecto es la construcción de una maqueta que mostrase la capacidad de Arduino para controlar un cruce semafórico.

Debido a la complejidad de la situación propuesta al ir incluyendo elementos, se consideró realizar la maqueta tipo panel con todos sus elementos soldados sobre una placa de circuito impreso virgen. Pues de otro modo la maqueta no tendría la gran robustez que se ha logrado con el denominado **panel de pruebas**. Así todos los componentes quedan soldados directamente a la placa y no hay elementos que se puedan dañar accidentalmente por su uso normal.

A su vez, se consideró el hecho de hacerla versátil. Por este motivo el panel puede utilizarse para probar gran cantidad de situaciones, tengan o no que ver con cruces de semáforos. Algunos ejemplos prácticos de ello se enuncian a continuación:

- Activar y desactivar por pulsación una secuencia de luces concreta.
- Contador de paso por barrera infrarroja, mostrando el resultado en el display.
- Iniciar una cuenta atrás con una pulsación que provoque la intermitencia de un led o conjunto de estos.

4.2 DISEÑO DEL ESQUEMÁTICO

En este apartado se comentan los criterios de diseño para el correcto funcionamiento del panel de pruebas.

Se aconseja utilizar resistencias asociadas en serie con los LEDs para reducir el voltaje proporcionado por los pines de Arduino. Así evitaremos la destrucción de los mismos.

La utilización de interrupciones con Arduino puede hacerse a veces complicado porque la placa es muy sensible a cualquier cambio de valor en los pines asignados a interrupciones (2 y 3). De hecho una interrupción puesta en modo LOW, es decir, que se active cuando el pin esté a masa es activada con sólo un pequeño trozo de cable suelto. Por ello el pulsador para los peatones es **normalmente cerrado** poniendo así el pin2 a una tensión constante de 3.3 voltios para fijar el estado HIGH.

También en la interrupción para el semáforo C se utiliza una entrada, cuando permanece inactiva, a nivel alto sacada del inversor.

La función del inversor en el panel es la de ajustar a nivel alto o bajo la referencia de voltaje sacada de la rama del **fotodiodo SFH 213 FA**. El **inversor** utilizado es de tecnología TTL (5 voltios) pero también podría usarse de tecnología MOS ya que su salida cuando está a nivel alto es de 3.3 voltios.

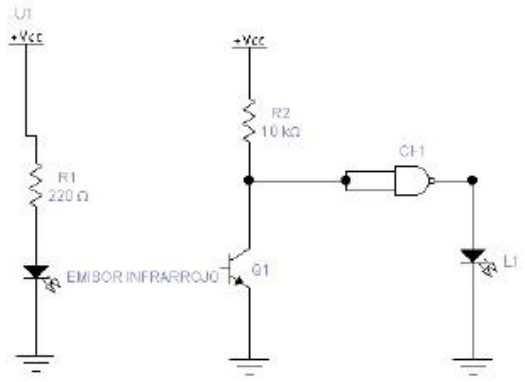


Figura 4.2.1 Circuito detector de infrarrojos

Como sensor de barrera se utiliza un fotodiodo (SFH 213 FA) que recibe una señal continua desde un diodo LED de infrarrojos. También forma parte del diseño un diodo amarillo de comprobación. Éste debe estar encendido mientras nada obstaculice el haz de infrarrojos. El diseño está basado en el circuito de la figura 4.2.1.

La circuitería para la barrera láser adaptada quedaría como se muestra en la figura 4.2.2.

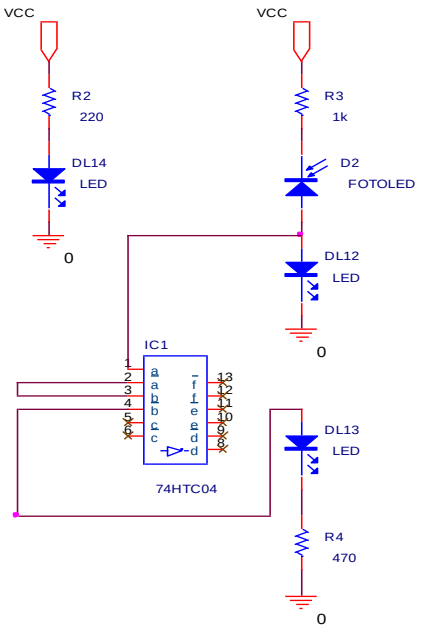


Figura 4.2.3 Barrera de infrarrojos

El circuito completo para el panel de pruebas realizado en Orcad se muestra en la figura 4.2.4. Como se puede observar consta de una regleta de 23 pines hembra para la conexión al panel de Arduino o cualquier otra tarjeta controladora de 3 a 6 voltios.

Los parts (dibujos que representan a un componente) correspondientes a la regleta, el pulsador y el 74HCT04 han sido creados para su uso en este circuito.

Las resistencias de protección de los siete segmentos del display han sido sustituidas en el diseño de las pistas de la placa de circuito impreso por una sola de 1K en el cátodo común del display. Esto se ha hecho por economizar en el trazado de pistas.

Las hojas de datos de los componentes menos estándar vienen recogidas en el **Anexo C**.

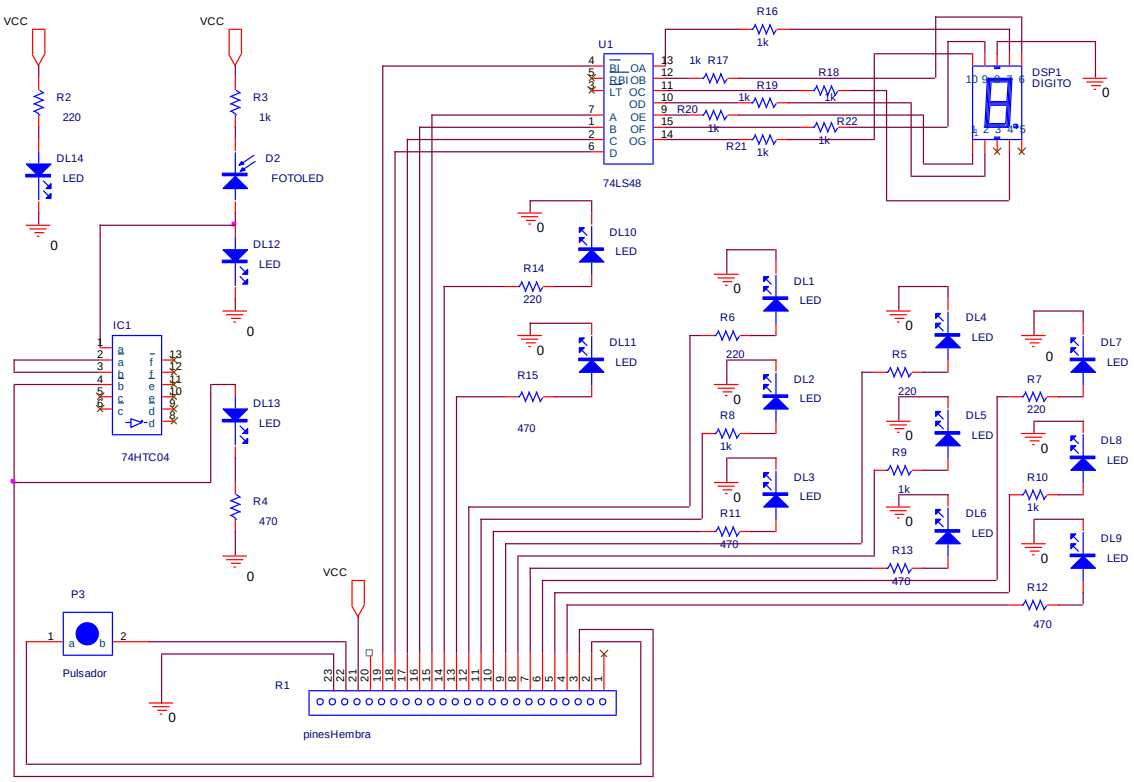


Figura 4.2.4 Circuito completo del panel de pruebas

4.3 DISEÑO DE LA PCB

Tras crear el esquemático de la figura 4.2.4 y asignar los footprints correspondientes al tamaño y forma real de cada componente, se pasa el diseño a Layout para crear la distribución de las pistas (Nets). Al borde de placa se le ha asignado el tamaño correspondiente de la placa virgen a fabricar.

Como se puede apreciar tras el trazado de las pistas en la placa ayudado de la herramienta Autoroute utilizamos dos capas (Layers) pues en una sólo no podemos trazar toda la PCB. Esto es debido a que algunas pistas se entrecruzan.

La placa virgen a fabricar posee una sola cara. Siendo así, la capa que aparece de color rojo (BOTTON) que evidentemente tiene menor número de pistas trazadas, se implementará a través de cableado por la misma cara que la capa azul (TOP).

El resultado final se puede observar en la figura 4.3.1. Luego seleccionamos la capa TOP (la de color azul) y obtenemos la imagen a imprimir (figura 4.3.2) para posteriormente pasar el tóner de la fotocopia a la superficie de cobre de la placa virgen, aplicando calor con una plancha. Después se repasa el dibujo sobre el cobre con un rotulador permanente y se sumerge en una disolución de ácido clorhídrico y agua oxigenada (acelerante) para la eliminación del cobre expuesto.

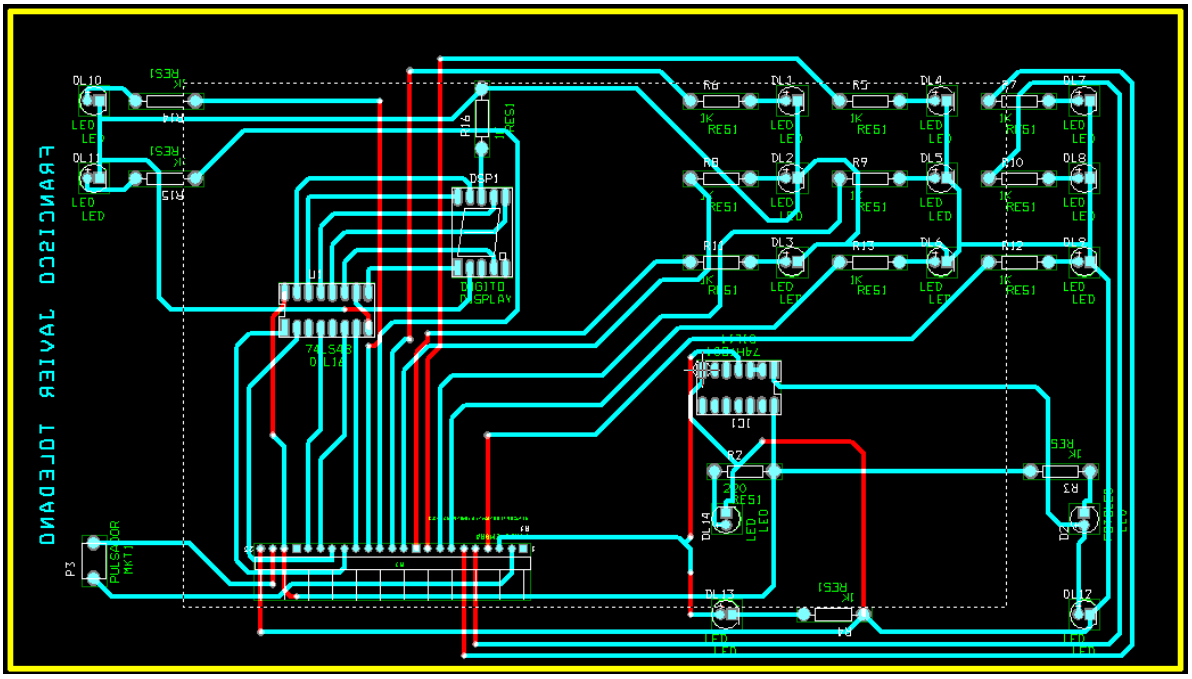


Figura 4.3.1 Pistas trazadas con Orcad Layout

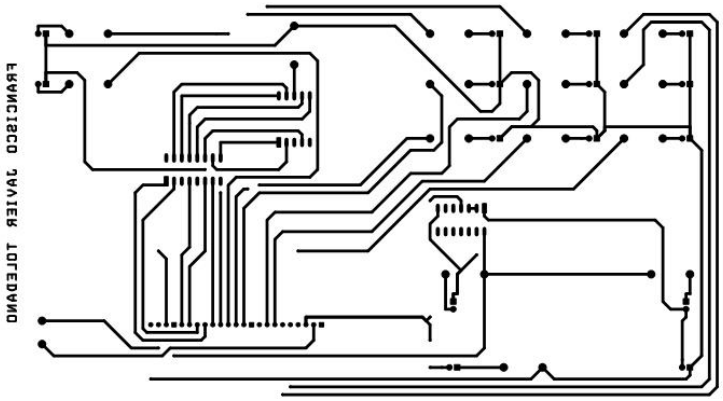


Figura 4.3.2 Capa TOP

4.4 USO DEL PANEL

El uso de este panel es sencillo. En la figura 4.4.1 la fila azul muestra que elemento electrónico está asociado a tal pin de la regleta hembra soldada a la placa de circuito impreso. El panel está diseñado para funcionar a valores de tensión de Arduino (5 voltios), pero hay una tolerancia de 0.6 voltios.

Arduino se debe conectar a la regleta con cable unifilar en el caso de no utilizar los conectores empleados para este proyecto.

Regleta de conexiones:

Numeración de pines en el panel de derecha a izquierda.
Elemento del panel correspondiente a la conexión.
Pin asignado en Arduino para el programa.
Nombre de las variables declaradas en el programa.

11	10	9	8	7	6	5	4	3	2	1
Amarillo A	Verde A	Rojo B	Amarillo B	Verde B	Rojo C	Amarillo C	Verde C	Int. Semáf C	Int. Peatones	
10	9	8	7	6	5	4	19	3	2	
Pin2	Pin3	Pin4	Pin5	Pin6	Pin7	Pin8	Pin9			
22	21	20	19	18	17	16	15	14	13	12
3.3 Voltios	5 Voltios		BI/RBO	A3	A2	A1	A0	Rojo Peatones	Verde Peatones	Rojo A
3.3 v	5 v		18	17	16	15	14	13	12	11
			PinA4	PinA3	PinA2	PinA1	PinA0	Pin10	Pin11	Pin1
										23
										GND
										GND

Figura 4.4.1 Regleta de conexiones

4.5 ETAPA DE ADAPTACIÓN DE NIVELES DE TENSIÓN

El panel de pruebas está diseñado para funcionar a valores de 5v. Por tanto si se desea hacer uso de éste con un autómata convencional, simplemente debemos añadir una etapa muy simple de optoacopladores para adaptar los 24 voltios del autómata a los 5 voltios del panel de pruebas y viceversa.

Para una sola señal el circuito básico quedaría como el de la figura 4.5.1.

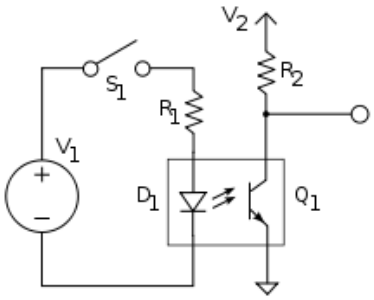


Figura 4.5.1 Adaptador de tensión todo/nada

Para controlar con Arduino varios actuadores o elementos eléctricos (relés) cuyo funcionamiento es superior a 5 voltios, podemos utilizar una configuración de optoacopladores muy simple como la mostrada en la figura 4.5.2.

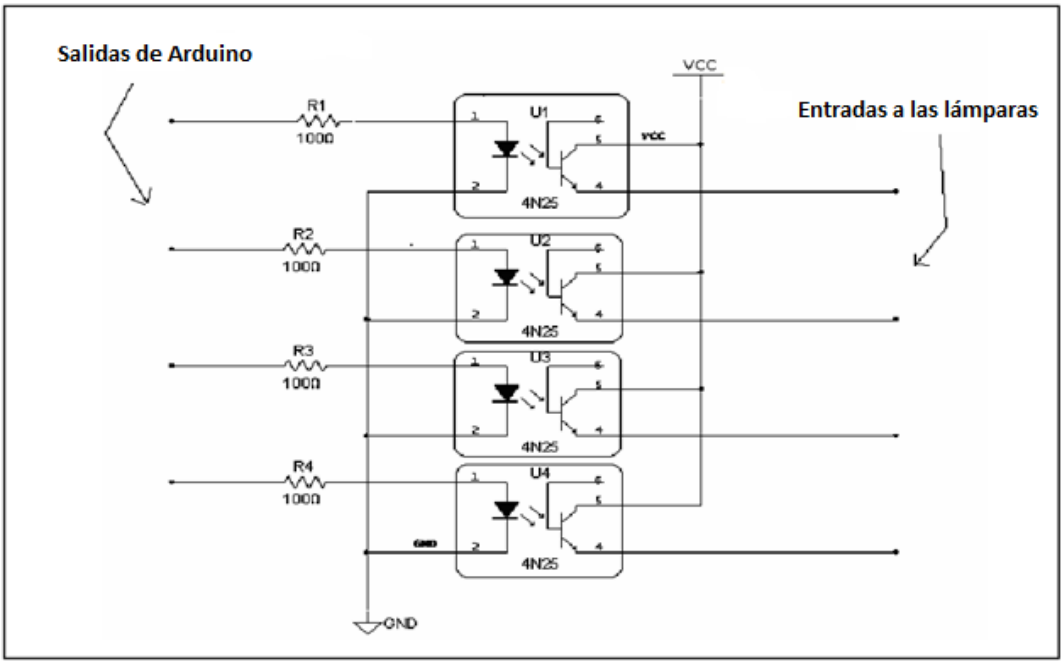


Figura 4.5.2 Configuración de optoacopladores

4.6 PRESUPUESTO DE EJECUCIÓN MATERIAL.

En la tabla 4.6.1 viene recogido el coste material de construcción del panel de pruebas y en la figura 4.6.1 el aspecto final del mismo.

CONCEPTO	CANTIDAD	PRECIO UNIDAD (IVA INCLUIDO)	TOTAL EN EUROS
Arduino Uno	1	25.84	25.84
Cable unifilar (1m)	1	0.42	0.42
Tablero A3	1	3.31	3.31
Placa Virgen A5 (1 cara)	1	3.63	3.63
Diodo LED 5mm	13	0.15	1.95
CI. 74LS48	1	2.50	2.50
CI. 74HCT	1	0.27	0.27
Display BCD/7segmentos	1	1.47	1.47
Pulsador NC	1	0.80	0.80
Resistencia 1/4W 5%	15	0.02	0.30
IR333_A	1	1.28	1.28
SFH-213 FA	1	0.98	0.98
Conmutador	1	1.00	1.00
Pila 9V	1	0.80	0.80
Tira pines hembra	1	0.46	0.46
Tira pines macho	2	1.59	3.18
Pegamento rápido	1	3.01	3.01
Tuercas y tornillos	10	0.18	1.80
Ácido clorhídrico	1	1.00	1.00
Rotulador permanente	1	1.70	1.70
Estaño	1	0.90	0.90
Agua oxigenada	2	0.60	1.20
Papel pegatina	1	0.20	0.20
Pasta para soldar	1	3.40	3.40
			61.40

Tabla 4.6.1 Presupuesto material del panel de pruebas

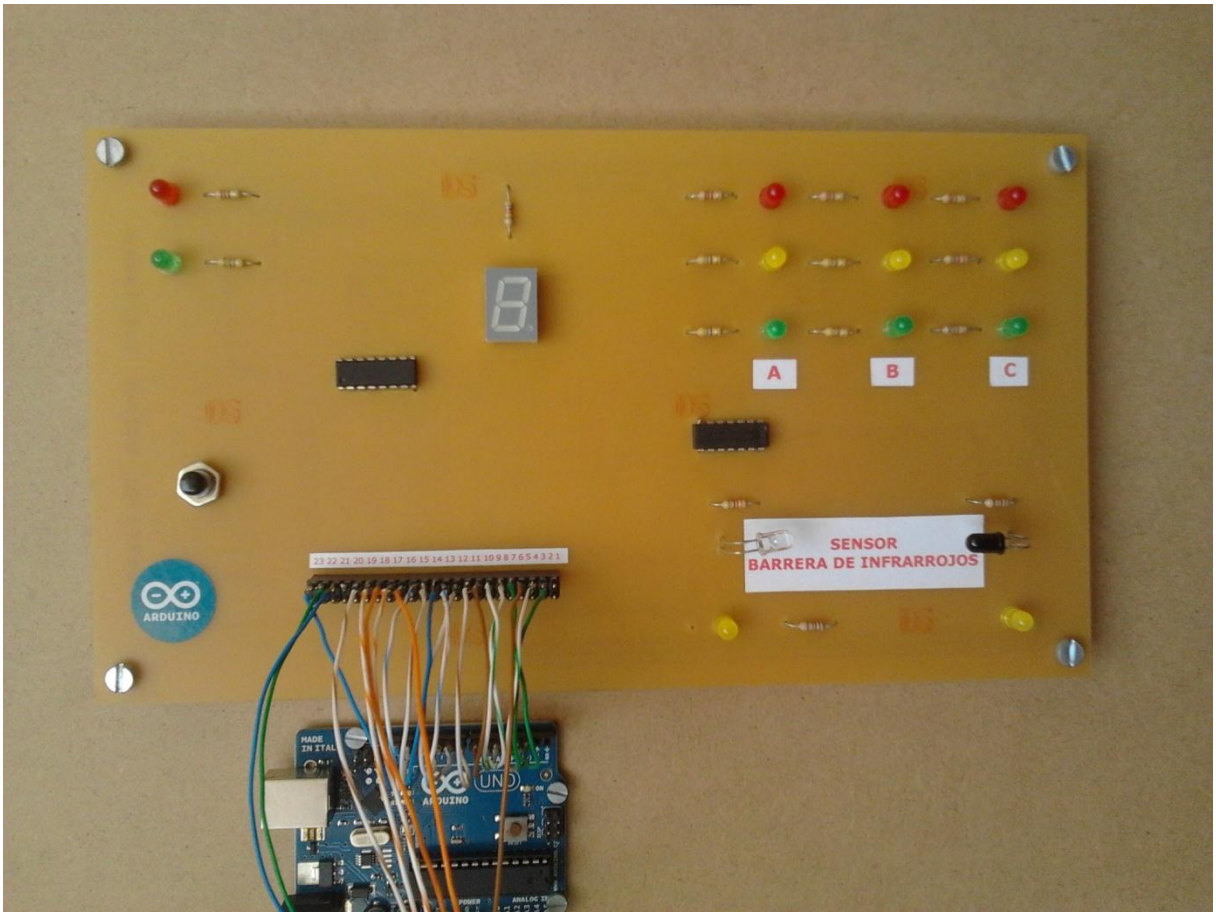


Figura 4.6.1 Aspecto final del panel de pruebas

Capítulo 5

SISTEMAS PARA EL CONTROL SEMAFÓRICO.

5.1 INTRODUCCIÓN

Para el control de cruces de semáforos en la actualidad se utilizan los llamados reguladores semafóricos. En este capítulo comparamos estos sistemas con uno en el que se utiliza Arduino como elemento controlador.

Formaremos un sistema centralizado usando la interfaz física RS-232 como vía de comunicación. Por ello deberemos usar Arduino Serial cuyas características se comentan en la página 34 de esta memoria.

Por último, acabaremos este capítulo con un presupuesto para ambos sistemas y así mostrar la rentabilidad de usar Arduino en lugar de un regulador semafórico.

5.2 SISTEMAS ACTUALES PARA EL CONTROL SEMAFÓRICO

En actualidad, el control de cruces de semáforos se realiza a través de reguladores semafóricos, que en la mayoría de los casos constan de un procesador y conexión a un ordenador central a través del protocolo TCP/IP como elementos más reseñables.

Estos reguladores específicos para esta función tienen una gran desventaja respecto a la utilización de tarjetas de control discretas, como Arduino, que radica en su elevado precio para realizar la misma función. En la mayoría de los casos no se utilizan muchas de las funciones de dichos productos. En la figura 5.2.1 se muestra el esquema de un regulador semafórico.

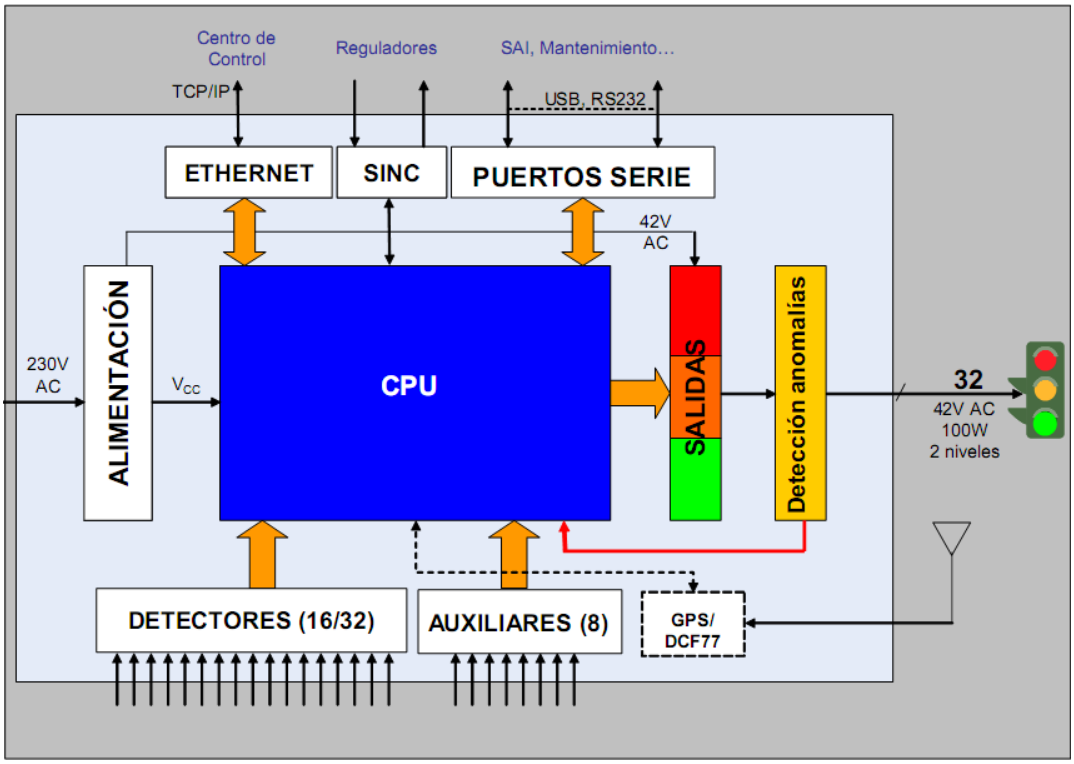


Figura 5.2.1 Esquema de un regulador semafórico

Tanto a nuestro sistema de control como al del regulador semafórico se aconseja añadir una fuente de alimentación ininterrumpida (SAI) para asegurar el abastecimiento de energía a los semáforos. Aunque en muchos proyectos se considera opcional. En la figura 5.2.2 vemos como se instalaría la SAI.

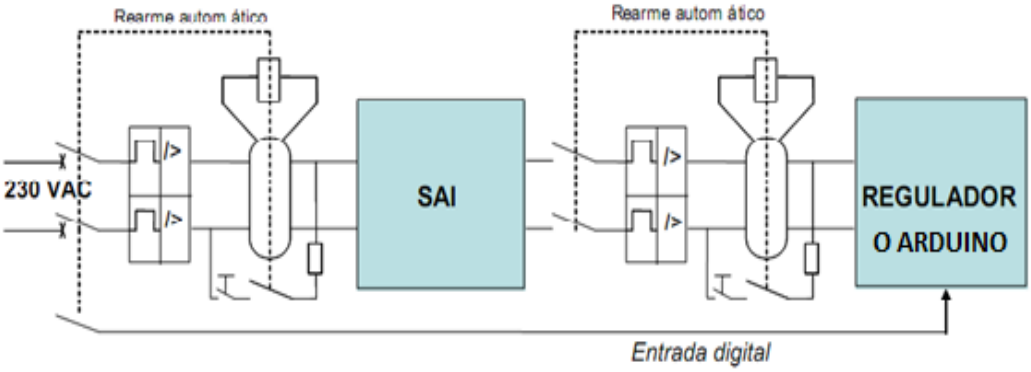


Figura 5.2.2 Sistema de alimentación interrumpida

Podemos ver a continuación un par de ejemplos de reguladores semafóricos obtenidos de proyectos reales, sobre cuyos presupuestos suelen oscilar estos sistemas:

- Ud. Regulador semafórico Electrónico sincronizable, coordinable y centralizable, capacidad de 8 programas con variación de ciclos,reparto y desfase realizado por microprocesador para 10 grupos semafóricos, limpieza de elementos, verificación de equipo y programación, incluso montaje-desmontaje, P.P. de pequeño material y medios auxiliares.

  5.978,17€

- Ud. regulador local electrónico dotado de microprocesador, autónomo y sincronizable y centralizable, con multiprogramas seleccionables, actuado por espiras con extensiones de tiempos, con módulo de detección y control de lámpara fundida, con salidas estáticas, incluso armario exterior y ocho grupos semafóricos de capacidad (mod. RD), programado y colocado.

  3.850,57€

5.3 SISTEMA CENTRALIZADO CON ARDUINO SERIAL

Vamos a coger el caso de dos grupos semafóricos y montar el sistema completo de control centralizado de los mismos usando Arduino Serial (Arduino Serial se programa exactamente igual que Arduino Uno). De forma que podamos reprogramar en tiempo real el funcionamiento de los semáforos a través de una interfaz creada en LabVIEW. La figura 5.3.1 muestra un esquema simplificado de los principales elementos necesitados para controlar dos cruces de semáforos desde un ordenador central.

Figura 5.3.1 Sistema centralizado con Arduino Serial

El sistema completo consta de los siguientes elementos:

- Arduinos Serial. Un inconveniente de este hardware es que lo suelen vender con la placa y los componentes sin soldar. Se recomienda como ejercicio didáctico por los fabricantes.

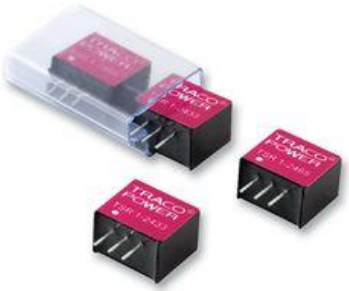
- Transmisores RS-232 de 1000m de alcance máximo. Son muy comunes y nos lo ofrecen gran cantidad de distribuidores. Algunos de ellos son:
  - www.planetronic.es
  - www.pixmania.com
  - www.mercamania.es
  - www.mayoristacable.es
  - www.conectalo.com

- Etapas de adaptación de niveles. Ya se ha hablado en la página 53 sobre ellas. La fabricación es fácil y muy económica.

- Cable conexión serie (db9).

- Fuente de alimentación para aplicaciones industriales, mejorando así su fiabilidad y de coste más reducido. Una opción sería http://www.electan.com/fuente-alimentacion-24v-cebek-p-2107.html

- Convertidor cc-cc 24/9v. Para la alimentación de Arduino. Un ejemplo sería: http://es.farnell.com/tracopower/tsr-1-2490/converter-dc-dc-24v-9v-1a-sip/dp/1696322



- SAI para garantizar alimentación a los semáforos en caso de problemas en el suministro eléctrico. Se recomienda pero se considera opcional en la mayoría de proyectos de instalación de semáforos. Los reguladores semafóricos no la integran.

5.4 COMPARACIÓN DE SISTEMAS

Podemos apreciar que el sistema basado en Arduino contiene más elementos para conexionar. Ello conlleva un aumento de la probabilidad de error por el mayor número de elementos, pero que en ningún caso es un problema para el correcto funcionamiento del sistema.

El regulador semafórico viene con unas rutinas ya programadas y listas para ser cargadas. En nuestro sistema también podemos desarrollar diferentes programas a la hora de comercializarlo, para que cualquier persona pueda reprogramar un grupo semafórico sin dificultad alguna.

En las tablas 5.4.1 y 5.4.2 vemos el coste de ambos sistemas incluyendo una SAI. En ningún caso se ha tenido en cuenta el sobrecoste consecuencia del posterior mantenimiento.

El coste del regulador semafórico se ha cogido de la base de datos proporcionada por el Instituto Valenciano de la Edificación.

Como se puede apreciar hay una diferencia de **2542.29€**. El resultado sería un sistema más complejo y menos integrado, pero sustancialmente más económico.

Concepto	Precio (IVA incluido)	Cantidad	Total
Arduino Serial	20.00	2	40.00
Transmisores RS-232 hasta 1000m	159.51	3	478.53
Etapa adaptación de niveles	10	2	20
Cable conexión serie (db9)	4.13	3	12.39
Fuente de alimentación	80.06	2	160.12
Convertidor cc-cc 24/9v	7.70	2	15.40
SAI 230vac (opcional)	39.83	2	79.65
			806.09€

Tabla 5.4.1 Presupuesto para dos grupos semafóricos usando Arduino Serial

Concepto	Precio (IVA incluido)	Cantidad	Total
Regulador semafórico	3308.55	1	3308.55
SAI 230vac (opcional)	39.83	1	39.83
			3348.38€

Tabla 5.4.2 Presupuesto para dos grupos semafóricos usando un Regulador Semafórico

Capítulo 6

Conclusión y líneas futuras.

6.1 CONCLUSIÓN

Gran parte del control electrónico en la actualidad se realiza a través de microcontroladores. Arduino es una plataforma microcontroladora que destaca en el mercado por las siguientes características:

- **Su reducido coste**. Cuando adquirimos una placa de Arduino, estamos pagando el coste material, el coste de producción, las ganancias del proveedor, el transporte y el IVA. Aún así, tenemos que en el mercado no hay ninguna plataforma microcontroladora que pueda rivalizar con Arduino en este aspecto.

- **Gran variedad de versiones de Arduino**. Como hemos visto en el apartado 2.5 y 2.6 de esta memoria de proyecto, podemos seleccionar el tipo de placa más acorde a nuestras exigencias.

- **Hardware libre**. En caso de necesitar alguna característica que no se encuentre en ninguna placa, el fabricante nos proporciona el esquema electrónico de la mayoría de placas en su web oficial pudiendo de este modo modificar lo que necesitemos.

- **Software libre**. El compilador que nos proporciona el fabricante es de código abierto. De este modo los usuarios de Arduino pueden modificar lo que deseen, adaptando así el programa a sus necesidades.

- **Fácil de programar**. Arduino nos proporciona un listado de funciones que vienen incluidas en el bootloader que nos facilitan la programación de su microcontrolador.

- **Fácil de adquirir.** Está muy extendido en el mercado.

La aplicación de Arduino en sistemas semafóricos de pequeñas poblaciones, así como en otros proyectos relacionados con el control de procesos industriales y en el área de las comunicaciones, resulta idóneo en las ocasiones en las que no se quiere comprar equipos muy caros de los que no se aprovecha todas sus funciones o cuando no se dispone del tiempo necesario de desarrollar un hardware específico para tal aplicación.

Con Arduino podemos implementar un sistema para ejecutar el control de semáforos tal y como los usuarios conocemos. En cambio, esto no es posible en España ni por lo general en países desarrollados, pues dicha actividad está regulada por una gran cantidad de **normas UNE** y alguna **norma CEI** (Comité Español de Iluminación) de grandes exigencias técnicas. En dichas normas, recogidas en la bibliografía de esta memoria, podemos ver que bastantes de los requerimientos **son imposibles de cumplir con cualquier placa Arduino.**

Concluimos, poniendo de manifiesto la no viabilidad del control de grupos semafóricos con Arduino.

6.2 FUTURAS LÍNEAS DE TRABAJO

Se proponen como posibles futuras líneas de trabajo las siguientes:

1. El control de ascensores resulta especialmente idóneo para ejecutarlo con Arduino.

2. La automatización de procesos, en donde no solo podemos realizar el control de actuadores y recibir señales todo/nada como entrada, sino que se puede utilizar para linealizar sensores, comunicarnos con un ordenador central o con otras unidades de control, variar la potencia de actuadores según se necesite,…

3. Mediante el uso de la shield Ethernet se consigue el control vía internet de lo que se desee. Una aplicación de ello sería su uso en domótica. Resulta relativamente sencillo controlar nuestro sistema de climatización conectando Arduino a los pulsadores del panel del climatizador. De este modo enfriamos o calentamos nuestra vivienda para que esté preparada antes de llegar.

4. Últimamente se ha puesto muy de moda integrar luces LED en las vestimentas de grupos de baile para enriquecer el espectáculo. Normalmente están siempre encendidas o se tiene sobre ellas el limitado control que proporciona un interruptor. Pues bien, usando Arduino BT o la shield Xbee se conseguiría un control de las tiras de LED que significarían una clara oportunidad de negocio.

BIBLIOGRAFÍA Y REFERENCIAS.

[1] Guía de usuario de Arduino.
Rafael Enríquez Herrador
Universidad de Córdoba
Noviembre de 2009

[2] Web oficial de Arduino.
http://www.arduino.cc/es

[3] Autómatas programables.
Josep Balcells y José Luis Romeral

[4] Fundamentos de informática. Programación en C.
Pedro María Alcover Garau
Universidad politécnica de Cartagena
Septiembre de 2006

[5] Regulador semafórico. Especificaciones técnicas y funcionales.
Ayuntamiento de Barcelona
Mayo de 2008

[6] Wikipedia.
http://es.wikipedia.org

[7] Memoria Plan E2. Cruce semafórico.
Damián Monreal Palencia
Ayuntamiento de Jumilla
Enero de 2010

[8] Semáforo de peatones.
José A. Rodríguez Mondéjar.
UPCO ICAI Departamento de Electrónica y Automática 1

[9] Proveedores de equipos electrónicos.
www.planetronic.es
www.pixmania.com
www.mercamania.es
www.mayoristacable.es
www.conectalo.com
www.electan.com
www.farnell.com

[10] UNE-HD 638 S1:2001
Sistemas de señalización del tráfico viario

[11] UNE 135401-1 EX
Equipamiento para la señalización vial – Reguladores de tráfico
Parte 1: Características funcionales

[12] UNE 135401-2 EX
Equipamiento para la señalización vial – Reguladores de tráfico
Parte 2: Métodos de prueba

[13] UNE 135401-3
Equipamiento para la señalización vial – Reguladores de tráfico
Parte 3: Características eléctricas

[14] UNE 135401-5 IN
Equipamiento para la señalización vial – Reguladores de tráfico
Parte 4: Protocolo de comunicaciones, tipo V

[15] UNE 135401-6
Equipamiento para la señalización vial – Reguladores de tráfico
Compatibilidad electromagnética

[16] UNE-EN 12675:2001
Semáforos. Requisitos funcionales de seguridad

[17] UNE-EN 50293:2001
Compatibilidad electromagnética
Sistemas de señalización del tráfico por carretera
Norma de producto

[18] UNE-EN 60068-2-64
Ensayos ambientales
Parte 2: Métodos de ensayo
Ensayo Fh: Vibración aleatoria de banda ancha (control digital) y guía

[19] EN 50102:1995
Grados de protección proporcionados por las envolventes de materiales eléctricos contra los impactos mecánicos externos (código IK)

[20] EN 60068-2-75
Ensayos ambientales.
Parte 2: Ensayos. Ensayo Eh: Ensayos de martillos

[21] EN 60259:1991
Grados de protección proporcionados por las envolventes (código IP)

[22] UNE 20324 Erratum
Grados de protección proporcionados por las envolventes (código IP)

[23] EN 60068-2-2:1993
Ensayos ambientales Parte 2: Ensayos Ensayo B: Calor seco

[24] EN 60068-2-1:1993
Ensayos ambientales Parte 2: Ensayos Ensayo B: Frío

[25] EN 60068-2-30:1999
Ensayos ambientales
Parte 2: Ensayos
Ensayo Db y guía: Ensayo cíclico de calor húmedo (ciclo de 12+12 horas)

[26] EN 60068-2-5:1999
Ensayos ambientales
Parte 2: Ensayos
Ensayo Sa: Radiación solar artificial al nivel de la superficie terrestre

[27] UNE 20460-5-54:1990
Instalaciones eléctricas en edificios. Elección e instalación de los materiales eléctricos. Puesta a tierra y conductores de protección

[28] CEI 60536
Clasificación de los equipos eléctricos y electrónicos respecto a la protección contra choques eléctricos

[29] CEI 60-1
Técnicas de ensayo de alta tensión
Parte 1: Definiciones y prescripciones generales relativas a los ensayos

[30] UNE-EN 61008-1:1996
Interruptores automáticos para actuar por corriente diferencial residual, sin dispositivo de protección contra sobreintensidades, para usos domésticos y análogos (ID)

[31] UNE-EN 55022
Equipos de tecnología de la información
Características de las perturbaciones radioeléctricas
Límites y métodos de medida

[32] UNE-EN 55014
Compatibilidad electromagnética
Requisitos para aparatos electrodomésticos, herramientas eléctricas y aparatos análogos
Parte 1: Emisión

[33] UNE-EN 61000-4-2
Compatibilidad electromagnética
Parte 4: Técnicas de ensayo y medida
Sección 2: Ensayos de inmunidad a las descargas electrostáticas
Norma básica de CEM

[34] UNE-EN 61000-4-3
Compatibilidad electromagnética
Parte 4-3: Técnicas de ensayo y medida
Ensayos de inmunidad a los campos electromagnéticos, radiados y de radiofrecuencia

[35] UNE-EN 61000-4-4
Compatibilidad electromagnética
Parte 4: Técnicas de ensayo y medida
Sección 4: Ensayos de inmunidad a los transitorios eléctricos rápidos en ráfagas

[36] UNE-EN 61000-4-5
Compatibilidad electromagnética
Parte 4: Técnicas de ensayo y medida
Sección 5: Ensayos de inmunidad a las ondas de choque

[37] UNE-EN 61000-4-6
Compatibilidad electromagnética
Parte 4: Técnicas de ensayo y medida
Sección 6: Inmunidad a las perturbaciones conducidas, inducidas por campos de radiofrecuencia

[38] UNE-EN 61000-4-8
Compatibilidad electromagnética
Parte 4: Técnicas de ensayo y medida
Sección 8: Ensayo de inmunidad a los campos magnéticos a la frecuencia industrial
Norma básica de CEM

[39] UNE 21308-1:1994
Ensayos en alta tensión.
Parte 1: definiciones y prescripciones generales relativas a los ensayos

[40] HD 588.1 S1:1991
High-voltage test techniques
Part 1: General definitions and test requirements

[41] UNE-EN 60950-1:2003
Equipos de tecnología de la información. Seguridad. Parte 1: Requisitos generales

[42] UNE-EN 61000-3-2:2001
Compatibilidad electromagnética (CEM). Parte 3-2: Límites.
Límites para las emisiones de corriente armónica (equipos con corriente de entrada <= 16 A por fase).

[43] UNE-EN 61000-3-3:1997
Compatibilidad electromagnética (CEM). Parte 3: Límites.
Sección 3: Limitación de las variaciones de tensión, fluctuaciones de
tensión y flicker en las redes públicas de suministro de baja tensión para equipos con corriente de entrada <= 16 A por fase y no sujetos a una conexión condicional.

[44] Ordenanza general del medio ambiente urbano.
Título III. Contaminación acústica

ANEXO A

TUTORIAL DE PLACAS.

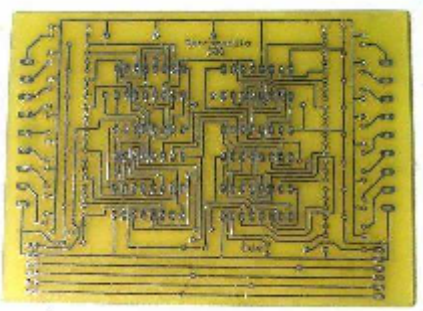
INTRODUCCIÓN

Hacer una placa de circuito impreso no tiene porque ser una tarea complicada. Quizás muchos de vosotros habéis renunciado a hacerlas tras haberos desanimado por múltiples razones, pero básicamente porque a priori parece muy difícil, sobre todo si habéis pensado en hacerlo de una forma profesional, usando insoladora y placa fotosensible. Eso está bien si hemos de hacer muchas placas, pero muchos de nosotros queremos hacer alguna de forma ocasional. Os garantizo que si seguís con detalle las indicaciones de este tutorial, con un poco de paciencia y práctica obtendréis unos excelentes resultados.

Mis primeras placas las hice con un rotulador permanente. Con este método preparé algunas placas sencillas. Pero en cuanto el circuito se complicaba algo o las pistas eran algo finas era inútil. Aún así, para hacer algunas placas con pocas pistas y cierta separación entre ellas, se consiguen buenos resultados. El resultado tampoco era muy profesional, pero funcionaba bien.

Cuando la red de pistas se complicó y quise preparar otras placas, lo del rotulador era inviable. Así que estuve una temporada con la idea de construirme una insoladora. La tarea no era sencilla, y el coste tampoco era pequeño. También necesitaba placa fotosensible y trabajar en penumbra un rato para el lavado adicional. Casualmente, navegando por internet, encontré el procedimiento de "transferencia de tóner". Me pareció fenomenal cambiar la insoladora por la plancha y la solución de sosa cáustica por agua.

He de deciros que fui escéptico al principio. En cualquier caso probar era barato. Y experimentar también. Dicho y hecho. Hice muchas pruebas hasta que conseguí los resultados que quería; placas perfectas. Incluso con pistas superfinas.
Echadle un vistazo a estas placas:



Podría poner muchas más, pero para muestra son suficientes. Todas están hechas con el método que os describo más adelante.
Parte de lo que necesitáis ya lo tenéis en casa, y el resto es fácil de conseguir. Lo más importante es el papel fotográfico. De este último hay muchos tipos y precios. El que yo os indico es con el que he obtenido los mejores resultados de los que he probado; tiene la particularidad que una vez planchado se desprende fácilmente del cobre al remojarlo. Los demás os quedarán pegados y al retirarlos os llevaréis también las pistas. Yo lo compré en Carrefour y me costó el sobre con 20 hojas 12€. Veinte hojas dan para muchas placas. Los hay mucho más baratos, pero usad éste. Os lo digo por experiencia.

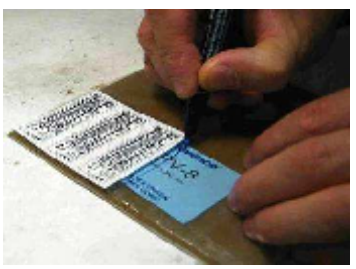
LOS MATERIALES

- Papel fotográfico glossy marca EPSON, ref. SO41126
- Placa de circuito impreso (sierra y lija de grano medio)
- Un plancha
- Un par de túpers
- Lija de metal fina
- Papel de cocina o un paño limpio
- Agua oxigenada
- Aguafuerte
- Tijera, alcohol, pinzas y un punzón de punta finito
- Un cepillo de dientes viejo
- Un rotulador permanente(indeleble)

Y alguna otra cosa más que ya os indicaré en el reportaje.

PREPARACIÓN DE LA PLACA

Lo primero que tenemos que hacer es recortar el circuito que previamente habremos impreso en una impresora láser. Recordad la importancia que tiene usar el papel indicado. Procurad también no tocar la zona impresa con los dedos; cogedla por los bordes. Cualquier grasilla que tengáis en los dedos pasará al papel y las pistas de esa zona no agarrarán bien en la placa. Tened en cuenta que el resulado final va a depender mucho de la de la calidad de la impresión. Cuanto mejor sea la impresión, mejor serán los resultados que obtendremos.

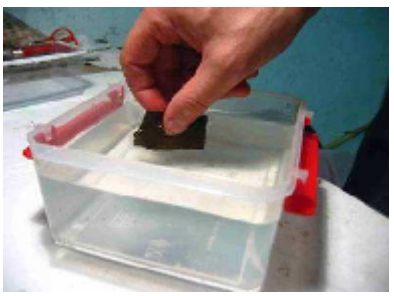


A continuación marcamos sobre la placa de circuito impreso la zona que posteriormente recortaremos con la sierra. Yo suelo usar placa de fibra de vidrio; es más cara que la placa de baquelita, y aunque es más dura y por lo tanto más difícil de cortar y taladrar, es más duradera, resistente y aguanta mucho mejor el calentamiento. Es para toda la vida.

Dejad algo de margen al marcar, ya que esto facilitará las tareas de planchado. Medio centímetro de más en cada lado está bien. Yo me quedé algo corto en esta placa.

Cortamos la placa guiándonos de las marcas. Para ello yo me ayudo con una sargenta para sujetarla a la mesa, es mucho más cómodo que sujetarla con la mano y además haréis un corte recto y seguro.

Ahora retiramos todas las rebabas que han quedado del corte; para ello frotamos la placa por las dos caras por cada uno de los bordes. La siguiente tarea es pulir ligeramente la superficie de la placa para que el tóner agarre mejor. Para ello, humedecéis un trozo de lija fina para metal en agua, mojáis la placa y la lijáis suavemente por toda la superficie.



Veréis que el cobre pierde su aspecto brillante y queda ligeramente rallado. Procurad pulir bien toda la superficie, especialmente por los laterales. Tendemos siempre a pulir el centro y descuidamos las esquinas.

Al acabar, la mojáis bien de nuevo y la secáis con un paño limpio o un papel de cocina. Para eliminar bien todas impurezas que pudieran quedar sobre la placa, apoyamos esta sobre la mesa le echamos unas gotas de alcohol isopropílico y lo limpiamos bien. Si no lo tenéis usad alcohol normal, es menos efectivo pero sirve.

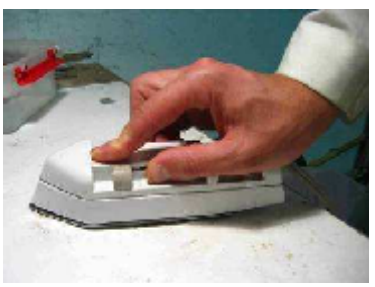

Yo suelo repetir esta operación un par de veces hasta que al pasar el trapo, éste sale completamente limpio.

A partir de ahora, no toquéis el cobre de la placa con los dedos, si necesitáis cogerla, hacerlo cuidado por los bordes.

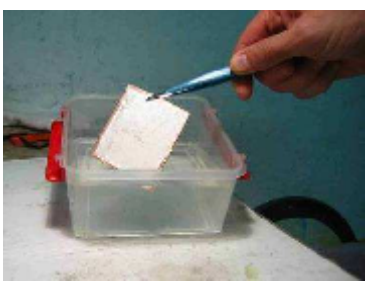
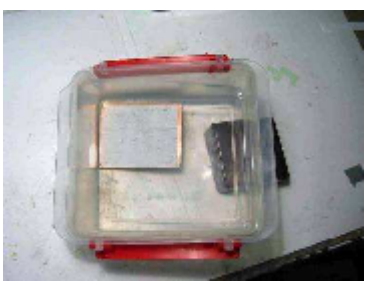
GRABADO DE LA PLACA

Terminada la fase de pulido y limpieza del cobre, pasamos ahora a la etapa de planchado, que fijará el tóner sobre la superficie de la placa. Primero colocamos la hoja recortada al principio boca abajo sobre el cobre, centradita.



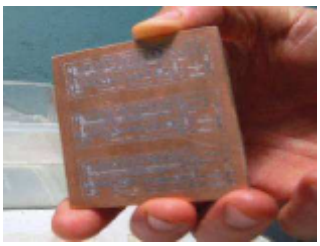


Colocad una tabla vieja debajo de la placa si hacéis la operación sobre una mesa de casa, ya que al calentarse la placa podría decolorarse o levantarse el barniz de la mesa.
A continuación y con la plancha bien caliente (posición de algodón y sin vapor) empezáis el planchado. Al principio hacedlo con cuidado, como en la foto del medio, sujetando el papel por un extremo y pasando la plancha por el otro. Enseguida el tonner empezará a desprenderse y pegará la hoja al cobre.
Es el momento de calentar bien la plaquita. Dejad la plancha un ratito encima, y después planchad con fuerza por toda la superficie. Hacedlo sin miedo.
Yo uso una vieja plancha de viaje, que es más manejable, pero la de casa sirve perfectamente. Si queréis evitar que la plancha se manche, poned un papel cocina entre la base de la plancha y la placa.

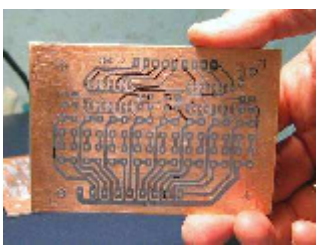



Terminado el planchado, con unas pinzas colocad la placa en agua, al contacto con ésta crepitará. Eso es buena señal. Dejadla descansar durante un rato.

El papel se desprenderá sin problemas en pocos minutos. Podéis ayudarle moviendo el recipiente haciendo olas, igual que durante el lavado. Debe despegarse sin ayuda. No lo forcéis. Si no es así dejadla un poco más a remojo. En cuanto seque observaréis que tiene pegado trocitos de papel, no os preocupéis, ahora los retiraremos. Si veis que el resultado es muy malo, no tiréis la placa. La lijáis bien para retirar todo el tóner y repetís el proceso; limpiáis bien con isopropílico la placa y la volvéis a grabar planchando otra hoja impresa con el circuito.



El papel que está encima de las pistas no nos molesta, pero el que se encuentra uniendo pistas distintas hay que eliminarlo. Para ello mojamos de nuevo la placa y con un cepillo de dientes húmedo frotamos la superficie. Si el tóner está bien fijado no se desprenderá. Moved el cepillo en la dirección de las pistas. Secadla y observad el resultado. A lo mejor tenéis que insistir. No tengáis prisa, hacedlo con cuidado, sobre todo cuanto más delgadas sean las pistas. Cercioraros bien de que no quede nada de papel entre las pistas, porque echaría a perder nuestra placa. Si queda algún trozo rebelde eliminadlo con un punzón muy fino. Haced esta operación con sumo cuidado para no dañar pistas colindantes.



No siempre el proceso sale a la perfección, como en la placa que estamos preparando. En ocasiones quedan algunos cortes entre pistas como en la placa de la izquierda. No os preocupéis. Con un rotulador indeleble las repasáis y listo.

LAVADO DE LA PLACA

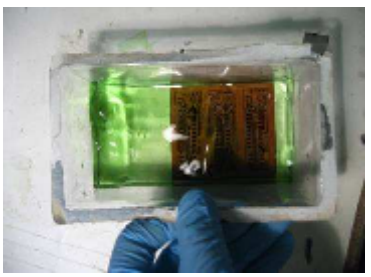
Pasamos a la fase de lavado. En ella sacaremos el cobre de la placa excepto de las zonas grabadas. Para ello emplearemos productos químicos: ácido clorhídrico y agua oxigenada. El primero de ellos es tóxico por inhalación o contacto y se emplea a nivel doméstico para limpieza. Se compra diluido y también se le llama aguafuerte, comercialmente Salfumant. Lo encontraréis en cualquier súper y es muy barato. El agua oxigenada del súper es la misma que la de la farmacia y más barata.

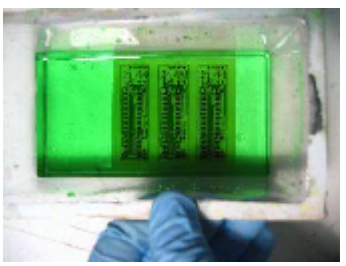
Tomad precauciones; realizad las operaciones siguientes en un lugar ventilado, poneros mascarilla de fieltro, guantes, una bata o ropa vieja y unas gafas plásticas. Cualquier descuido puede daros un disgusto. El ácido clorhídrico se come el metal, así que sacad conclusiones. Tampoco quiero asustaros. No vamos a fabricar una bomba atómica, pero mejor se precavidos.

Esto es todo lo que necesitamos; aguafuerte, agua oxigenada y un recipiente plástico en el que quepa la placa. Tened preparado otro con agua corriente. También usaremos un vaso plástico para

medir las cantidades de los líquidos. Le hacéis unas marcas con un rotulador, ya que ambos se emplean a partes iguales.


Medimos cantidades iguales de ambos líquidos y los echamos con cuidado en el recipiente plástico. Seguidamente colocamos la placa y veréis como el conjunto toma una tonalidad verde, por efecto de la reacción química. No hace falta llenar el recipiente, basta que la placa nade en un par de dedos de líquido. Para acelerar el proceso, y acabar antes moved con cuidado el recipiente, haciendo olas. Hacedlo despacio para que no desborde y no mojar la mesa.



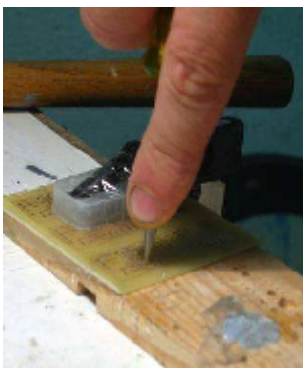


Esta operación dura sobre unos cinco minutos, aunque dependerá de la temperatura de los líquidos, la cantidad que echéis, la calidad de los productos, lo que mováis el túper…
Observad en la secuencia anterior como se va yendo el cobre en la foto central, y en la última ya ha desaparecido por completo.

Retiráis la placa y la colocáis en el recipiente que teníais preparado con agua limpia. Los productos químicos usados no los tiréis de golpe por el fregadero, echadlos diluidos con agua abundante poco a poco.

Con la misma lija que empleasteis para pulir el cobre, lijáis la placa. Veréis como se van descubriendo las pistas. Lijad y lavad la placa varias veces hasta que no quede resto de tóner. No os quitéis los guantes hasta acabar. El tóner es muy sucio. Para mí, este es el momento más agradecido del trabajo, porque van apareciendo las pistas poco a poco. Es un momento mágico que me llena de satisfacción.

Mirad la placa a contraluz y comprobad que todas las pistas están bien. Si tenéis alguna duda sobre la continuidad de alguna usad un polímetro para aseguraros. Podéis hacerlo también puenteando con una bombillita y una pila.

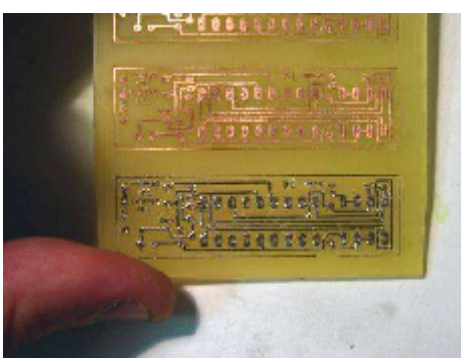



La placa está prácticamente lista; falta taladrarla, para lo que, previamente haremos unas guías con un punzón donde van los agujeros, para que al taladrar la broca no resbale. Para mi es la tarea más tediosa, pero no os la saltéis, si no los agujeros no irán donde deben.
Yo hago todos los agujeros con una broca de 0.7mm (la mayoría de los componentes usan este diámetro) y luego agrando a 1mm(conectores, puentes…) o 1,5mm(bornas) los que lo necesitan.

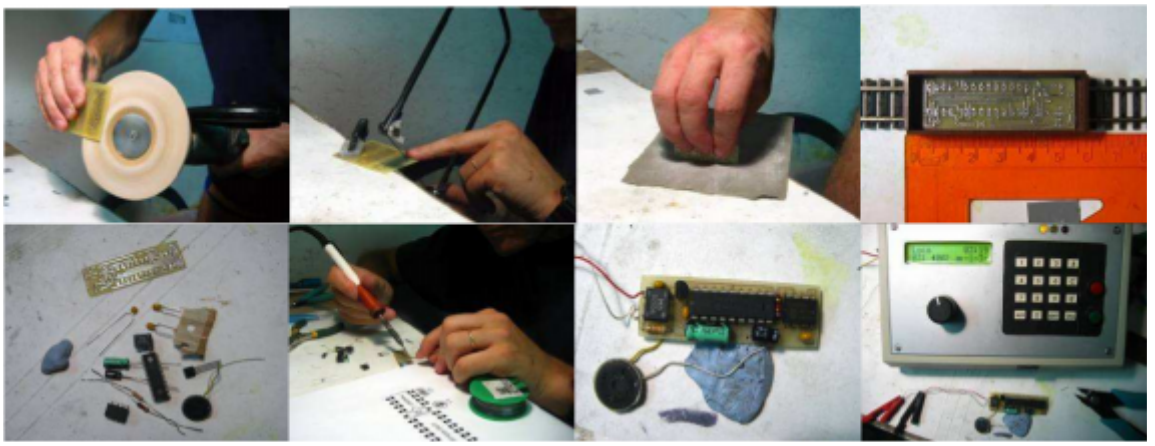
ESTAÑADO DE LA PLACA

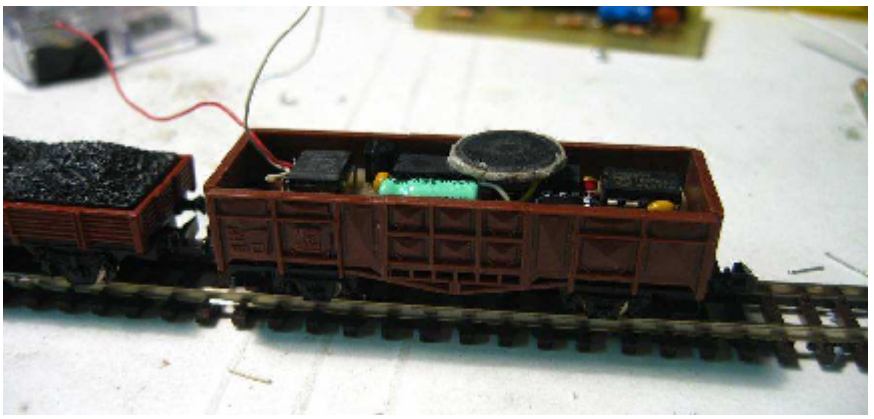

La siguiente operación es el estañado. Aunque no es imprescindible, facilitará enormemente la soldadura y además protegerá el cobre de la oxidación. Previamente barnizáis bien el cobre con Flux(una resina que se vende en tiendas de electrónica, barata), yo lo hacía antes con pincel, hasta que descubrí que con el dedo se acaba antes. Luego vais pasando el soldador con la punta ligeramente estañada por todas las pistas y quedarán marcadas con una preciosa capa plateada. Al acabar limpiáis el líquido que desprende la operación con un paño.

MONTAJE DE COMPONENTES



Hemos preparado tres placas iguales (que son tres decodificadores de sonido caseros para una locomotora) pero como podéis ver, solo he trabajado para la soldadura la última. Le rebajamos un poco el grosor, recortamos con la sierra, lijamos los bordes, preparamos los componentes, soldamos con cuidado, y probamos el montaje. Ahora a disfrutar de nuestro trabajo. Nos lo merecemos.





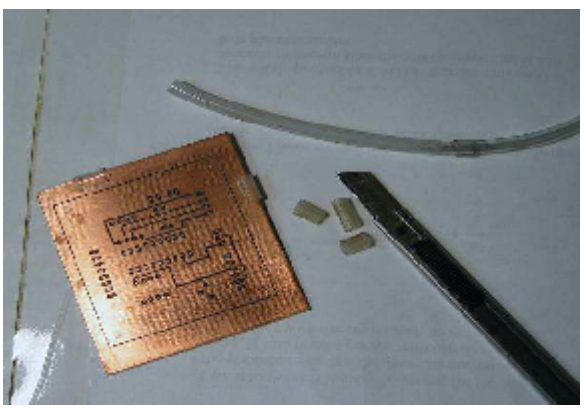
PLACAS DE DOBLE CARA

Este procedimiento también lo podéis emplear para hacer placas a doble cara. Es algo más laborioso y requiere más cuidado, pero también está a vuestro alcance.

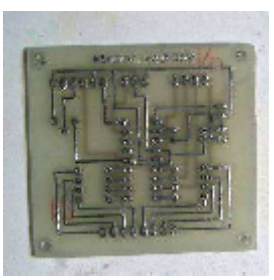
En primer lugar, sobre la hoja impresa con el circuito marcáis cuatro agujeros que os servirán de guía, los mismos en la cara de arriba y abajo. Coged los cuatro más esquinados, hacedlo con un punzón muy fino y centrad los agujeros muy bien. Recordad lo importante que es no tocar el papel con las manos.

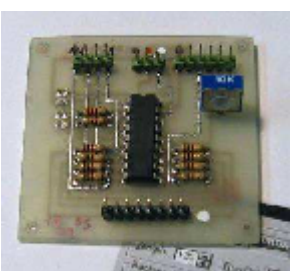
Colocad boca abajo sobre la placa(ya pulida y limpia por las dos caras como se indicó para placas sencillas) una de las caras impresas, sujetadla con unos trocitos de celo para que no se mueva y marcad con un punzón sobre la placa los agujeritos que practicasteis antes. Ahora, retirad con cuidado los celos y la hojita impresa.

Taladrad sin inclinar la broca con cuidado la placa por los puntos marcados. Utilizadlas como guía para colocar la hoja impresa ayudándoos con unos alfileres, sujetadla con celo, le dais la vuelta a la placa y repetís el proceso por la otra cara(alinear con los alfileres y sujetar con celo). Si habéis hecho el proceso con cuidado, las dos hojas impresas irán perfectamente enfrentadas. Ahora planchad bien por los dos lados como ya se indicó, poniendo entre la plancha y la placa un papel de cocina o un paño limpio, ya que el pegamento celo al calentarse manchará la plancha. Remojáis la placa un rato y retiráis el papel con cuidado.



He preparado dos placas, iguales y por uno de los lados he tenido que retocar alguna pista con el con el rotulador. Recortamos una de ellas. Para el lavado le colocáis a la placa en los laterales unos trocitos de macarrón plástico(lo podéis encontrar en las tiendas de animales; se emplean para la ventilación de las peceras) que cortaréis con el cúter en sentido longitudinal. Esto evitará que el toner de la placa por la zona inferior se deteriore y permitirá que se lave correctamente. Nos falta eliminar el cobre de las zonas sin tonner, lavando la placa como ya se indicó. Echad algo más de líquido, ya que hay más cobre que retirar. También tardará algo más en salir.





En la foto de la izquierda tenéis la placa lista, ya perforada y estañada por la parte de abajo. A la derecha, la placa montada, los componentes colocados en la parte de arriba. Este montaje es una sencilla central digital (MiniDcc) para manejar cuatro locomotoras en una maqueta de trenes.

ANEXO B

PROGRAMAS DESARROLLADOS.

B.1 Primer programa. Un semáforo.

```
int pin1 =5;   //declara pin1 asignado al pin número 5 de Arduino .
int pin2 =6;
int pin3 =7;

void setup()
{
 pinMode(pin1, OUTPUT);   //activa  el pin número 5 como salida.
 pinMode(pin2, OUTPUT);
 pinMode(pin3, OUTPUT);
 Serial.begin(9600);
}

void loop()
{
 verd();
 amar();
 rojo();
}

void verd()
{
 digitalWrite(pin3, HIGH);   //función  específica de Arduino que establece el pin designado a
 //valor alto o bajo.
 delay(10000);   //función de temporización. El argumento se escribe en milisegundos.
 digitalWrite(pin3, LOW);
}

void amar()
{
 digitalWrite(pin2, HIGH);
 delay(2000);
 digitalWrite(pin2, LOW);
}

void rojo()
{
 digitalWrite(pin1, HIGH);
 delay(10000);
 digitalWrite(pin1, LOW);
}
```

B.2. Segundo programa. Interrupción externa.

```
int pin=8;
volatile int x=0;

void setup()
{
  pinMode(pin,OUTPUT);
  attachInterrupt(0,pulso,LOW);  //función para activar y declarar una interrupción.
}

void loop()
{
  if (x==1){
  digitalWrite(pin,LOW);
  x=0;}
  else digitalWrite(pin,HIGH);

}

void pulso()
{
  x=1;  // esta función es llamada cuando se detecta una interrupción, cuyo resultado en este
//caso es la asignación del valor 1 a la variable x.
}
```

B.3. Tercer programa. Uso de la interrupción 0.

```
int pin1 =11;
int pin2 =9;
int pin3 =8;
int pin4 =7;
int pin5 =6;
int pin6 =5;
int pin7 =4;
int pin8 =3;
int pin9 =1;
int pin12=12;
int pin13=13;
volatile int x=0;

void setup()
{
   attachInterrupt(0,pulso,CHANGE);
 pinMode(pin1, OUTPUT);
 pinMode(pin2, OUTPUT);
 pinMode(pin3, OUTPUT);
 pinMode(pin4, OUTPUT);
```

```
 pinMode(pin5, OUTPUT);
 pinMode(pin6, OUTPUT);
 pinMode(pin7, OUTPUT);
 pinMode(pin8, OUTPUT);
 pinMode(pin9, OUTPUT);
 pinMode(pin12,OUTPUT);
 pinMode(pin13,OUTPUT);
 pinMode(2,INPUT);
 Serial.begin(9600);
}

void loop()
{
if(x==2)
          peatonal();
          else verd();
if(x==1)
          peatonal();
          else amar();
if(x==1)
          peatonal();
          else rojo();
if(x==1)
          peatonal();
          else verd1();
if(x==1)
          peatonal();
          else amar1();
if(x==1)
          peatonal();
          else rojo1();
if(x==1)
          peatonal();
          else verd2();
if(x==1)
          peatonal();
          else amar2();
if(x==1)
          peatonal();
          else rojo2();
}

void pulso()
{
  x=1;
}

void peatonal()
{
 digitalWrite(pin7,HIGH);
 digitalWrite(pin1,HIGH);
 digitalWrite(pin4,HIGH);
```

```
 digitalWrite(pin12,LOW);
 delay(1000);
 digitalWrite(pin13,HIGH);
 delay(10000);
 digitalWrite(pin13,LOW);
 digitalWrite(pin12,HIGH);
 delay(1000);
 x=0;
}

void verd()
{
  digitalWrite(pin12,HIGH);
 digitalWrite(pin3, HIGH);
  digitalWrite(pin7, HIGH);
   digitalWrite(pin4, HIGH);
    digitalWrite(pin1, LOW);
 delay(10000);
 digitalWrite(pin3, LOW);
}

void amar()
{
 digitalWrite(pin2, HIGH);
 delay(2000);
 digitalWrite(pin2, LOW);

}

void rojo()
{
 digitalWrite(pin1, HIGH);
 delay(1000);

}

void verd1()
{
   digitalWrite(pin4, LOW);
 digitalWrite(pin6, HIGH);
 delay(10000);
 digitalWrite(pin6, LOW);
}

void amar1()
{
 digitalWrite(pin5, HIGH);
 delay(2000);
 digitalWrite(pin5, LOW);
}

void rojo1()
```

```
{
 digitalWrite(pin4, HIGH);
 delay(1000);
}

void verd2()
{
   digitalWrite(pin7, LOW);
 digitalWrite(pin9, HIGH);
 delay(10000);
 digitalWrite(pin9, LOW);
}

void amar2()
{
 digitalWrite(pin8, HIGH);
 delay(2000);
 digitalWrite(pin8, LOW);
}

void rojo2()
{
 digitalWrite(pin7, HIGH);
 delay(1000);
 digitalWrite(pin7, LOW);
}
```

B.4. Cuarto programa. Programación cuenta atrás para el decodificador BCD a 7 segmentos.

```
int pin1 =11;
int pin2 =9;
int pin3 =8;
int pin4 =7;
int pin5 =6;
int pin6 =5;
int pin7 =4;
int pin8 =3;
int pin9 =10;
int pin12=12;
int pin13=13;
volatile int x=0;
int pinA0=14;  // Pin de entrada al decodificador para el bit de menor peso.
int pinA1=15;
int pinA2=16;
int pinA3=17;
int pinA4=18;

void setup()
{
```

```
  Serial.begin(9600);
  attachInterrupt(0,pulso,CHANGE);
 pinMode(pin1, OUTPUT);
 pinMode(pin2, OUTPUT);
 pinMode(pin3, OUTPUT);
 pinMode(pin4, OUTPUT);
 pinMode(pin5, OUTPUT);
 pinMode(pin6, OUTPUT);
 pinMode(pin7, OUTPUT);
 pinMode(pin8, OUTPUT);
 pinMode(pin9, OUTPUT);
 pinMode(pin12,OUTPUT);
 pinMode(pin13,OUTPUT);
pinMode(pinA0,OUTPUT);
pinMode(pinA1,OUTPUT);
pinMode(pinA2,OUTPUT);
pinMode(pinA3,OUTPUT);
pinMode(pinA4,OUTPUT);
}

void loop()
{
if(x==1)
          peatonal();
          else {
  interrupts();
 verd();
 amar();
}
if(x==1)
          peatonal();
          else
 rojo();
if(x==1)
          peatonal();
          else {
 verd1();
 amar1();
}
if(x==1)
          peatonal();
          else
 rojo1();
if(x==1)
          peatonal();
          else {
 verd2();
 amar2();
}
if(x==1)
          peatonal();
          else {
```

```
 rojo2();
  noInterrupts();
}
}

void pulso()
{
   x=1;
}

void peatonal()
{
 digitalWrite(pin7,HIGH);
 digitalWrite(pin1,HIGH);
 digitalWrite(pin4,HIGH);
  delay(1000);
 digitalWrite(pin12,LOW);
 digitalWrite(pin13,HIGH);

digitalWrite(pinA4,HIGH); // activa la entrada BI/RBO que enciende el display a través del
//decodificador.
digitalWrite(pinA0,HIGH);
digitalWrite(pinA3,HIGH);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,HIGH);
digitalWrite(pinA2,HIGH);
digitalWrite(pinA3,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,HIGH);
digitalWrite(pinA2,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA4,LOW);
```

```
 digitalWrite(pin13,LOW);
 digitalWrite(pin12,HIGH);
  delay(1000);
   x=0;
}

void verd()
{
 digitalWrite(pin12,HIGH);
 digitalWrite(pin3, HIGH);
 digitalWrite(pin7, HIGH);
 digitalWrite(pin4, HIGH);
 digitalWrite(pin1, LOW);
  delay(15000);
 digitalWrite(pin3, LOW);
}

void amar()
{
 digitalWrite(pin2, HIGH);
  delay(3000);
 digitalWrite(pin2, LOW);

}

void rojo()
{
 digitalWrite(pin1, HIGH);
  delay(2000);

}

void verd1()
{
 digitalWrite(pin4, LOW);
 digitalWrite(pin6, HIGH);
  delay(15000);
 digitalWrite(pin6, LOW);
}

void amar1()
{
 digitalWrite(pin5, HIGH);
  delay(3000);
 digitalWrite(pin5, LOW);
}

void rojo1()
{
 digitalWrite(pin4, HIGH);
  delay(2000);
}
```

```
void verd2()
{
 digitalWrite(pin7, LOW);
 digitalWrite(pin9, HIGH);
  delay(15000);
 digitalWrite(pin9, LOW);
}

void amar2()
{
 digitalWrite(pin8, HIGH);
  delay(3000);
 digitalWrite(pin8, LOW);
}

void rojo2()
{
 digitalWrite(pin7, HIGH);
  delay(2000);
 digitalWrite(pin7, LOW);
}
```

B.5. Programa final. Añadimos la interrupción 1.

```
int pin1 =11;
int pin2 =10;
int pin3 =9;
int pin4 =8;
int pin5 =7;
int pin6 =6;
int pin7 =5;
int pin8 =4;
int pin9 =19;
int pin10=13;
int pin11=12;
volatile int x=0;
volatile int y=0;
int pinA0=14;
int pinA1=15;
int pinA2=16;
int pinA3=17;
int pinA4=18;

void setup()
{
   Serial.begin(9600);
   attachInterrupt(0,pulso,LOW);
   attachInterrupt(1,sensor,LOW);
 pinMode(pin1, OUTPUT);
```

```
 pinMode(pin2, OUTPUT);
 pinMode(pin3, OUTPUT);
 pinMode(pin4, OUTPUT);
 pinMode(pin5, OUTPUT);
 pinMode(pin6, OUTPUT);
 pinMode(pin7, OUTPUT);
 pinMode(pin8, OUTPUT);
 pinMode(pin9, OUTPUT);
 pinMode(pin10,OUTPUT);
 pinMode(pin11,OUTPUT);
pinMode(pinA0,OUTPUT);
pinMode(pinA1,OUTPUT);
pinMode(pinA2,OUTPUT);
pinMode(pinA3,OUTPUT);
pinMode(pinA4,OUTPUT);
}

void loop()
{
  interrupts();  // activa las interrupciones.
 verd();
 amar();

if(x==1&&y==0)
          peatonal();
          else
 rojo();
if(x==1&&y==0)
          peatonal();
          else {
 verd1();
 amar1();
}
if(x==1&&y==0)
          peatonal();
          else
 rojo1();
if(x==1&&y==0)
          peatonal();
          else if (y==1) {
 verd2();
 amar2();
}
if(x==1&&y==0)
          peatonal();
          else {
 rojo2();
  noInterrupts(); // desactiva las interrupciones.
}
}

void pulso()
```

```
{
  x=1;
}

void sensor()
{
  y=1;
}

void peatonal()
{
 digitalWrite(pin7,HIGH);
 digitalWrite(pin1,HIGH);
 digitalWrite(pin4,HIGH);
  delay(1000);
 digitalWrite(pin10,LOW);
 digitalWrite(pin11,HIGH);

digitalWrite(pinA4,HIGH);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA3,HIGH);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,HIGH);
digitalWrite(pinA2,HIGH);
digitalWrite(pinA3,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,HIGH);
digitalWrite(pinA2,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA0,HIGH);
digitalWrite(pinA1,LOW);
delay(1000);
digitalWrite(pinA0,LOW);
delay(1000);
digitalWrite(pinA4,LOW);


 digitalWrite(pin11,LOW);
 digitalWrite(pin10,HIGH);
```

```
  delay(1000);
   x=0;
}

void verd()
{
 digitalWrite(pin10,HIGH);
 digitalWrite(pin3, HIGH);
 digitalWrite(pin7, HIGH);
 digitalWrite(pin4, HIGH);
 digitalWrite(pin1, LOW);
  delay(15000);
 digitalWrite(pin3, LOW);
}

void amar()
{
 digitalWrite(pin2, HIGH);
  delay(3000);
 digitalWrite(pin2, LOW);

}

void rojo()
{
 digitalWrite(pin1, HIGH);
  delay(2000);

}

void verd1()
{
 digitalWrite(pin4, LOW);
 digitalWrite(pin6, HIGH);
  delay(15000);
 digitalWrite(pin6, LOW);
}

void amar1()
{
 digitalWrite(pin5, HIGH);
  delay(3000);
 digitalWrite(pin5, LOW);
}

void rojo1()
{
 digitalWrite(pin4, HIGH);
  delay(2000);
}

void verd2()
```

```
{
 digitalWrite(pin7, LOW);
 digitalWrite(pin9, HIGH);
  delay(15000);
 digitalWrite(pin9, LOW);
}

void amar2()
{
 digitalWrite(pin8, HIGH);
  delay(3000);
 digitalWrite(pin8, LOW);
 y=0;
}

void rojo2()
{
 digitalWrite(pin7, HIGH);
  delay(2000);
 digitalWrite(pin7, LOW);
}
```

ANEXO C

DATASHEETS EMPLEADOS.

National Semiconductor

January 1992

DM74LS48
BCD to 7-Segment Decoder

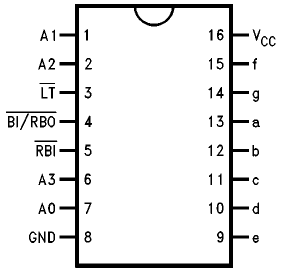
General Description

The 'LS48 translates four lines of BCD (8421) input data into the 7-segment numeral code and provides seven corresponding outputs having pull-up resistors, as opposed to totem pole pull-ups. These outputs can serve as logic signals, with a HIGH output corresponding to a lighted lamp segment, or can provide a 1.3 mA base current to npn lamp driver transistors. Auxiliary inputs provide lamp test, blanking and cascadable zero-suppression functions.

The 'LS48 decodes the input data in the pattern indicated in the Truth Table and the segment identification illustration.

Connection Diagram

Dual-In-Line Package



TL/F/10172–1

Order Number DM74LS48M or DM74LS48N
See NS Package Number M16A or N16E

Absolute Maximum Ratings (Note)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM74LS48			Units
		Min	Nom	Max	
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			V
V_{IL}	Low Level Input Voltage			0.8	V
I_{OH}	High Level Output Current			−50	μA
I_{OL}	Low Level Output Current			6.0	mA
T_A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} Min, I_{OH} = Max, V_{IL} = Max	2.4			V
I_{OFF}	Output High Current Segment Outputs	V_{CC} = Min, V_O = 0.85V	−1.3			mA
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min			0.5	V
		I_{OL} = 2.0 mA, V_{CC} = Min			0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V			20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.4	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max, V_O = 0V at $\overline{BI}/\overline{RBO}$ (Note 2)	−0.3		−2	mA
I_{CCH}	Supply Current	V_{CC} = Max, V_{IN} = 4.5V			38	mA

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	C_L = 15 pF		Units
		Min	Max	
t_{PLH} t_{PHL}	Propagation Delay Time A_n to a–g		100 100	ns
t_{PLH} t_{PHL}	Propagation Delay Time $\overline{RBI}$ to a–f		100 100	ns

Note: $\overline{LT}$ = HIGH, A_0–A_3 = HIGH.

Numerical Designations—Resultant Displays

TL/F/10172–4

Truth Table

Decimal Or Function	Inputs						Outputs							
	$\overline{LT}$	$\overline{RBI}$	A_3	A_2	A_1	A_0	$\overline{BI/RBO}$	a	b	c	d	e	f	g
0 (Note 1)	H	H	L	L	L	L	H	H	H	H	H	H	H	L
1 (Note 1)	H	X	L	L	L	H	H	L	H	H	L	L	L	L
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L
$\overline{BI}$ (Note 2)	X	X	X	X	X	X	L	L	L	L	L	L	L	L
$\overline{RBI}$ (Note 3)	H	L	L	L	L	L	L	L	L	L	L	L	L	L
$\overline{LT}$ (Note 4)	L	X	X	X	X	X	H	H	H	H	H	H	H	H

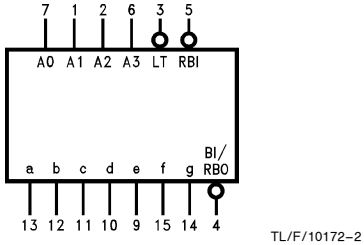
Note 1: $\overline{BI/RBO}$ is wired-AND logic serving as blanking input ($\overline{BI}$) and/or ripple-blanking output ($\overline{RBO}$). The blanking out ($\overline{BI}$) must be open or held at a HIGH level when output functions 0 through 15 are desired, and ripple-blanking input ($\overline{RBI}$) must be open or at a HIGH level if blanking of a decimal 0 is not desired. X = input may be HIGH or LOW.

Note 2: When a LOW level is applied to the blanking input (forced condition) all segment outputs go to a LOW level, regardless of the state of any other input condition.

Note 3: When ripple-blanking input ($\overline{RBI}$) and inputs A_0, A_1, A_2, and A_3 are at LOW level, with the lamp test input at HIGH level, all segment outputs go to a LOW level and the ripple-blanking output ($\overline{RBO}$) goes to a LOW level (response condition).

Note 4: When the blanking input/ripple-blanking output ($\overline{BI/RBO}$) is open or held at a HIGH level, and a LOW level is applied to lamp test input, all segment outputs go to a HIGH level.

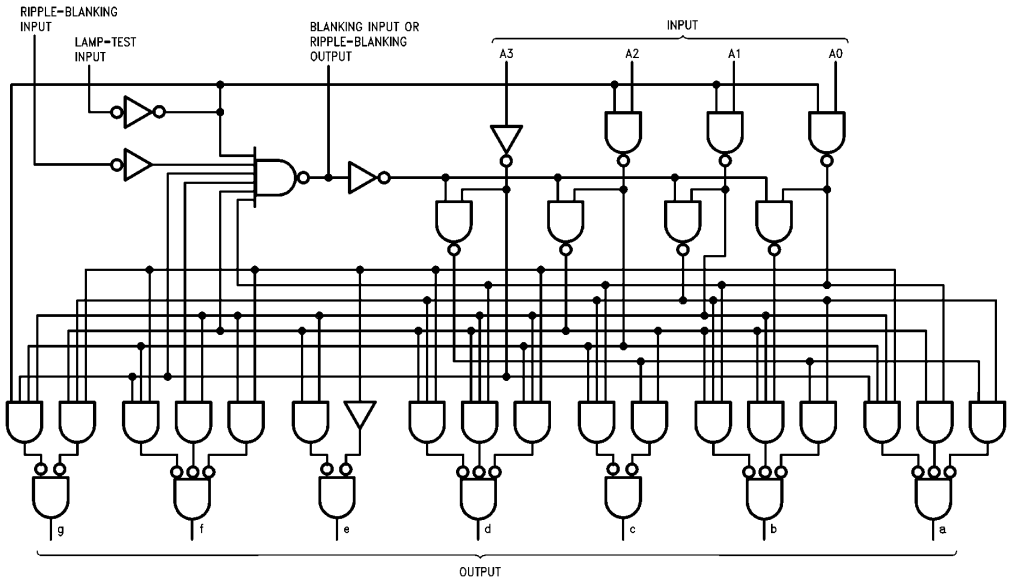
Logic Symbol



TL/F/10172–2

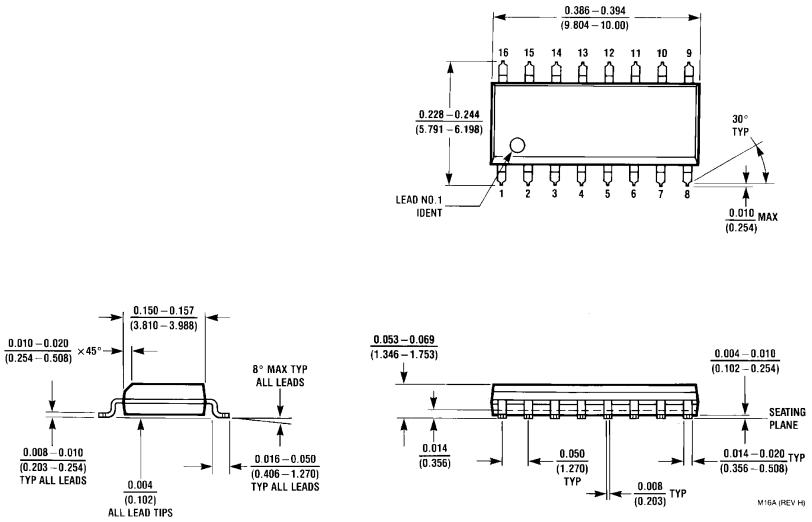
V_{CC} = Pin 16
GND = Pin 8

Logic Diagram



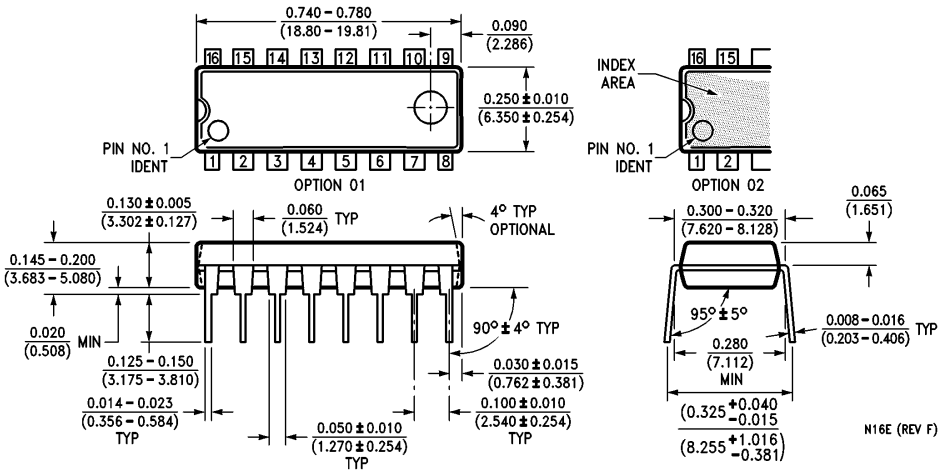
TL/F/10172–3

Physical Dimensions inches (millimeters)



16-Lead Small Outline Molded Package (M)
Order Number DM74LS48M
NS Package Number M16A

Physical Dimensions inches (millimeters) (Continued)



16-Lead Molded Dual-In-Line Package (N)
Order Number DM74LS48N
NS Package Number N16E

LIFE SUPPORT POLICY

NATIONAL'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 NATIONAL 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 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 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.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

FAIRCHILD SEMICONDUCTOR™

February 1984
Revised January 2005

MM74HCT04
Hex Inverter

General Description

The MM74HCT04 is a logic function fabricated by using advanced silicon-gate CMOS technology which provides the inherent benefits of CMOS - low quiescent power and wide power supply range. This device is input and output characteristic as well as pin-out compatible with standard 74LS logic families. The MM74HCT04, triple buffered, hex inverters, features low power dissipation and fast switching times. All inputs are protected from static discharge by internal diodes to V_{CC} and ground.

MM74HCT devices are intended to interface between TTL and NMOS components and standard CMOS devices. These parts are also plug-in replacements for LS-TTL devices and can be used to reduce power consumption in existing designs.

Features

- TTL, LS pin-out and threshold compatible
- Fast switching: t_{PLH}, t_{PHL}=12 ns (typ)
- Low power: 10 μW at DC, 3.7 mW at 5 MHz
- High fanout: ≥ 10 LS loads
- Inverting, triple buffered

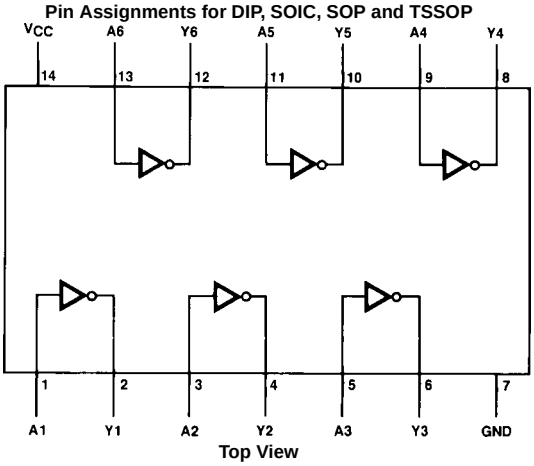
Ordering Code:

Order Number	Package Number	Package Description
MM74HCT04M	M14A	14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
MM74HCT04SJ	M14D	Pb-Free 14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
MM74HCT04MTC	MTC14	14-Lead Thin Shrink Small Outline Package (TSSOP), JEDEC MO-153, 4.4mm Wide
MM74HCT04N	N14A	14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
MM74HCT04N_NL	N14A	Pb-Free 14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.
Pb-Free package per JEDEC J-STD-020B.

Connection Diagram

Pin Assignments for DIP, SOIC, SOP and TSSOP



Top View

Absolute Maximum Ratings(Note 1)

(Note 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	
(Note 3)	600 mW
S.O. Package only	500 mW
Lead Temperature (T_L)	
(Soldering 10 seconds)	260°C

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage			
(V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temperature Range (T_A)	−40	+85	°C
Input Rise or Fall Times			
(t_r, t_f)		500	ns

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C.

DC Electrical Characteristics

V_{CC} = 5V ±10% (unless otherwise specified)

Symbol	Parameter	Conditions	T_A = 25°C	T_A = −40 to 85°C	T_A = −55 to 125°C	Units
			Typ	Guaranteed Limits		
V_{IH}	Minimum HIGH Level Input Voltage			2.0	2.0	2.0	V
V_{IL}	Maximum LOW Level Input Voltage			0.8	0.8	0.8	V
V_{OH}	Minimum HIGH Level Output Voltage	V_{IN} = V_{IL}					
		$\|I_{OUT}\|$ = 20 μA	V_{CC}	V_{CC} − 0.1	V_{CC} − 0.1	V_{CC} − 0.1	V
		$\|I_{OUT}\|$ = 4.0 mA, V_{CC} = 4.5V	4.2	3.98	3.84	3.7	V
		$\|I_{OUT}\|$ = 4.8 mA, V_{CC} = 5.5V	5.2	4.98	4.84	4.7	V
V_{OL}	Maximum LOW Level Voltage	V_{IN} = V_{IH}					
		$\|I_{OUT}\|$ = 20 μA	0	0.1	0.1	0.1	V
		$\|I_{OUT}\|$ = 4.0 mA, V_{CC} = 4.5V	0.2	0.26	0.33	0.4	V
		$\|I_{OUT}\|$ = 4.8 mA, V_{CC} = 5.5V	0.2	0.26	0.33	0.4	V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND, V_{IH} or V_{IL}		±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent Supply Current	V_{IN} = V_{CC} or GND		2.0	20	40	μA
		I_{OUT} = 0 μA					
		V_{IN} = 2.4V or 0.5V (Note 4)		0.3	0.4	0.5	mA

Note 4: This is measured per input with all other inputs held at V_{CC} or ground.

AC Electrical Characteristics

V_{CC} = 5.0V, $t_r = t_f$ = 6 ns C_L = 15 pF, T_A = 25°C (unless otherwise noted)

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PLH}, t_{PHL}	Maximum Propagation Delay		10	18	ns

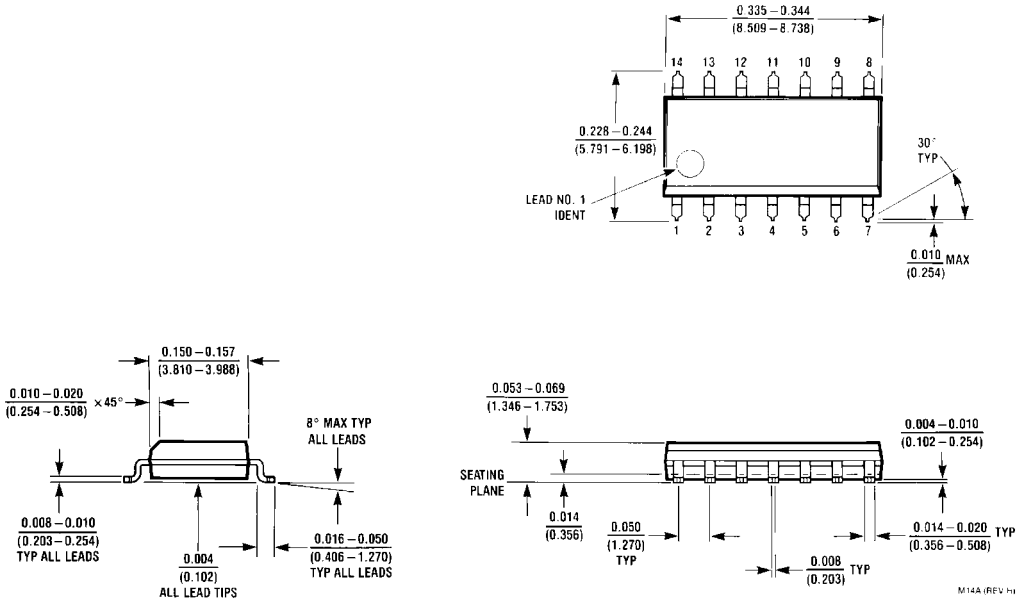
AC Electrical Characteristics

V_{CC} = 5.0V ±10%, $t_r = t_f$ = 6 ns, C_L = 50 pF (unless otherwise noted)

Symbol	Parameter	Conditions	T_A = 25°C	T_A = −40 to 85°C	T_A = −55 to 125°C		Units
			Typ	Guaranteed Limits			
t_{PLH}, t_{PHL}	Maximum Propagation Delay		14	20	25	30	ns
t_{THL}, t_{TLH}	Maximum Output Rise & Fall Time		8	15	19	22	ns
C_{PD}	Power Dissipation Capacitance	(Note 5)	20				pF
C_{IN}	Input Capacitance		5	10	10	10	pF

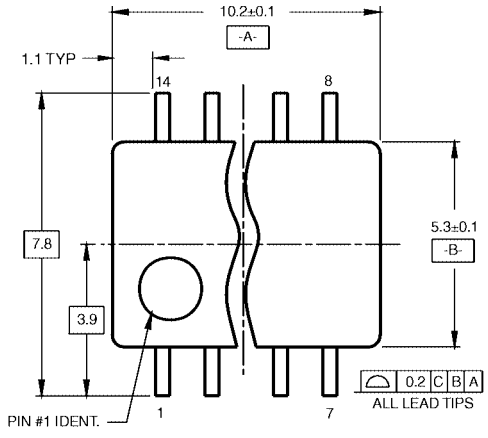
Note 5: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

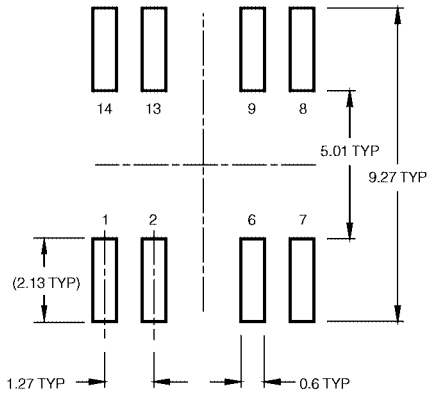
Physical Dimensions inches (millimeters) unless otherwise noted



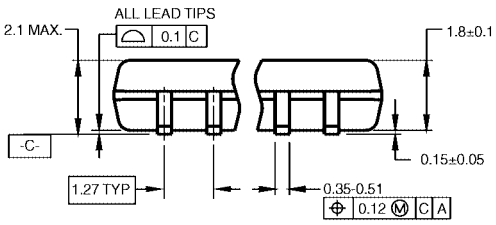
14-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150" Narrow
Package Number M14A

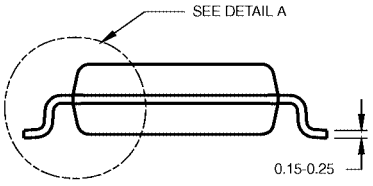
Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



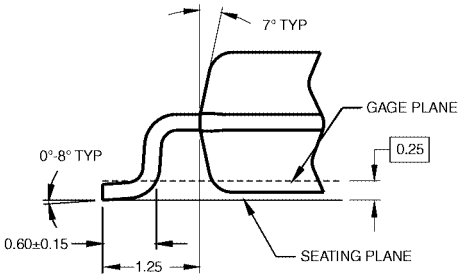


LAND PATTERN RECOMMENDATION





DIMENSIONS ARE IN MILLIMETERS



DETAIL A

NOTES:

A. CONFORMS TO EIAJ EDR-7320 REGISTRATION, ESTABLISHED IN DECEMBER, 1998.

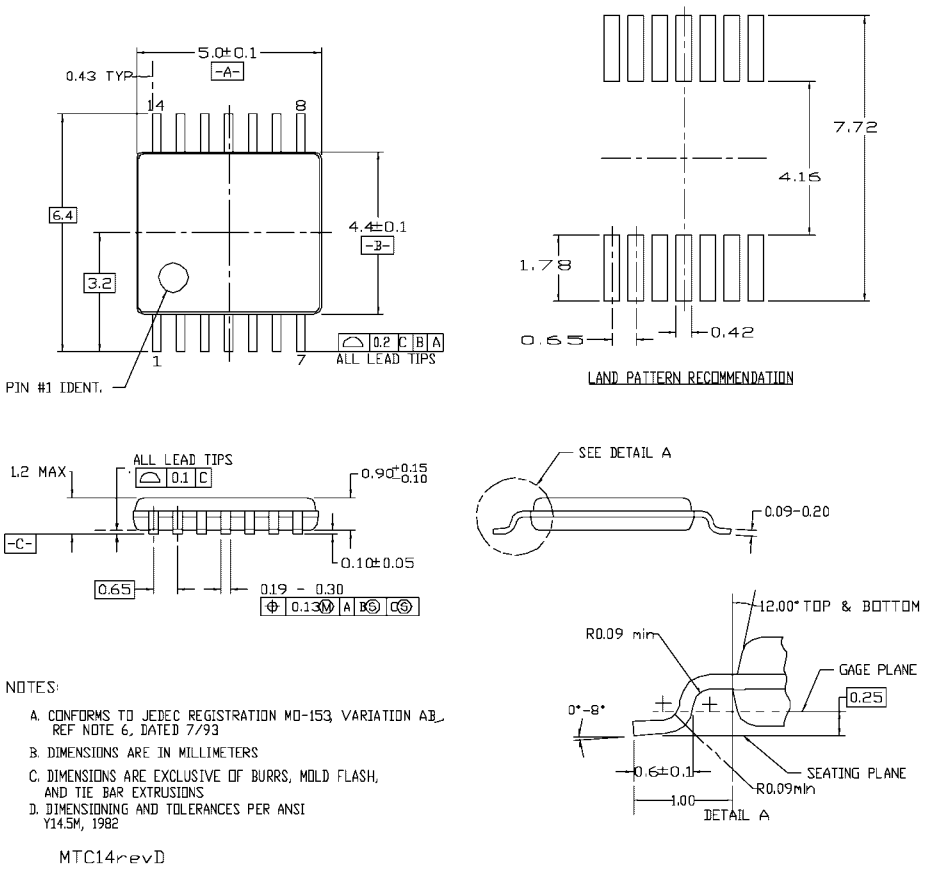
B. DIMENSIONS ARE IN MILLIMETERS.

C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS.

M14DRevB1

Pb-Free 14-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
Package Number M14D

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



NOTES:

A. CONFORMS TO JEDEC REGISTRATION MO-153, VARIATION AB, REF NOTE 6, DATED 7/93

B. DIMENSIONS ARE IN MILLIMETERS

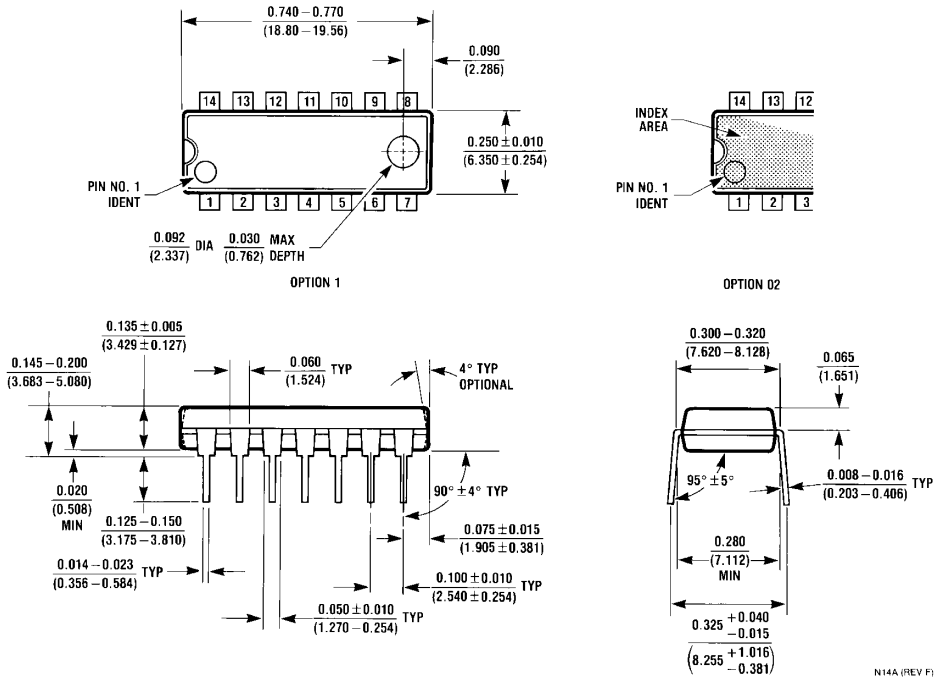
C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS

D. DIMENSIONING AND TOLERANCES PER ANSI Y14.5M, 1982

MTC14revD

14-Lead Thin Shrink Small Outline Package (TSSOP), JEDEC MO-153, 4.4mm Wide
Package Number MTC14

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



14-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
Package Number N14A

Technical Data Sheet

5mm Infrared LED , T-1 3/4

IR333-A

Features

- High reliability
- High radiant intensity
- Peak wavelength λp=940nm
- 2.54mm Lead spacing
- Low forward voltage
- Pb free
- The product itself will remain within RoHS compliant version.

Descriptions

- EVERLIGHT'S Infrared Emitting Diode(IR333-A) is a high intensity diode , molded in a blue transparent plastic package.
- The device is spectrally matched with phototransistor , photodiode and infrared receiver module.

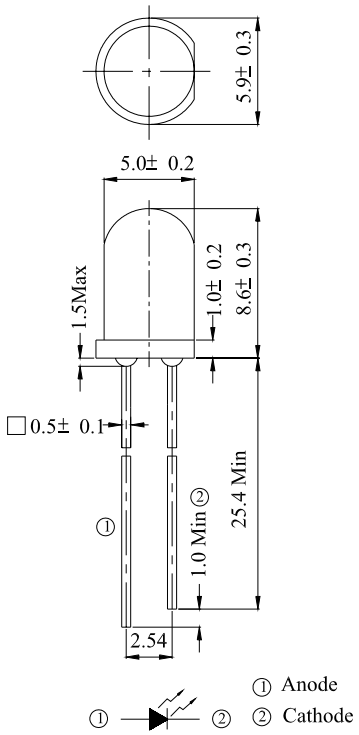
Applications

- Free air transmission system
- Infrared remote control units with high power requirement
- Smoke detector
- Infrared applied system

Device Selection Guide

LED Part No.	Chip Material	Lens Color
IR	GaAlAs	Blue

Package Dimensions



Notes: 1.All dimensions are in millimeters
2.Tolerances unless dimensions ±0.25mm

Absolute Maximum Ratings (Ta=25℃)

Parameter	Symbol	Rating	Units
Continuous Forward Current	I_F	100	mA
Peak Forward Current	I_{FP}	1.0	A
Reverse Voltage	V_R	5	V
Operating Temperature	T_{opr}	-40 ~ +85	℃
Storage Temperature	T_{stg}	-40 ~ +85	℃
Soldering Temperature	T_{sol}	260	℃
Power Dissipation at(or below) 25℃ Free Air Temperature	P_d	150	mW

Notes: *1:I_{FP} Conditions--Pulse Width≦100 μs and Duty≦1%.
*2:Soldering time≦5 seconds.

Electro-Optical Characteristics (Ta=25℃)

Parameter	Symbol	Condition	Min.	Typ.	Max.	Units
Radiant Intensity	Ee	I_F=20mA	7.8	20	--	mW/sr
		I_F=100mA Pulse Width≦100 μs ,Duty≦1%	--	85	--	
		I_F=1A Pulse Width≦100 μs ,Duty≦1%.	--	750	--	
Peak Wavelength	λp	I_F=20mA	--	940	--	nm
Spectral Bandwidth	Δλ	I_F=20mA	--	45	--	nm
Forward Voltage	V_F	I_F=20mA		1.2	1.5	V
		I_F=100mA Pulse Width≦100 μs ,Duty≦1%	--	1.4	1.8	
		I_F=1A Pulse Width≦100 μs ,Duty≦1%.	--	2.6	4.0	
Reverse Current	I_R	V_R=5V	--	--	10	μA
View Angle	2θ1/2	I_F=20mA	--	20	--	deg

Rank

Condition：I_F=20mA

Unit：mW/sr

Bin Number	M	N	P	Q
Min	7.80	11.0	15.0	21.0
Max	12.5	17.6	24.0	34.0

Typical Electro-Optical Characteristics Curves

Fig.1 Forward Current vs. Ambient Temperature

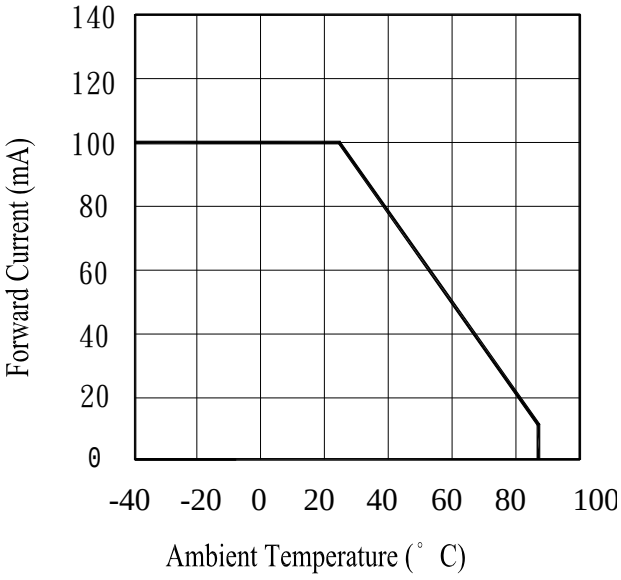


Fig.2 Spectral Distribution

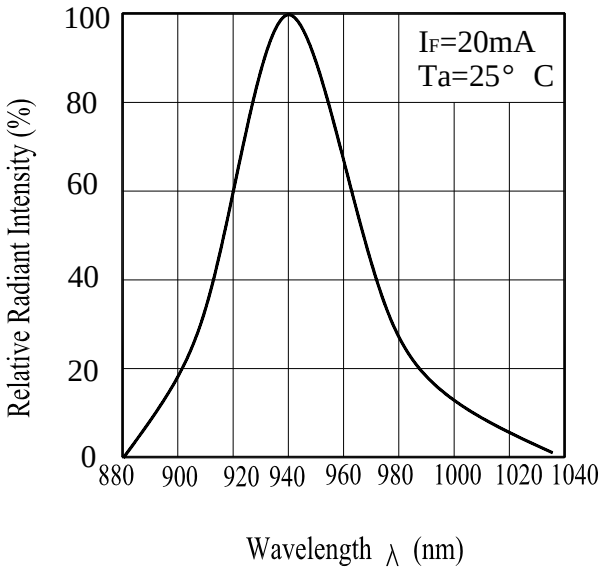


Fig.3 Peak Emission Wavelength Ambient Temperature

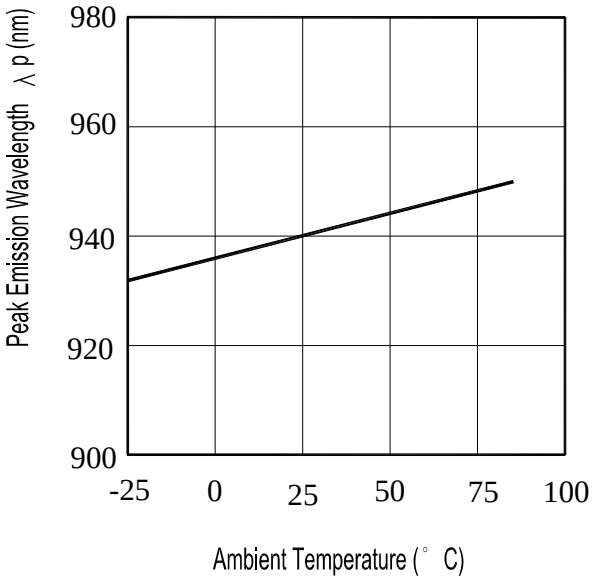
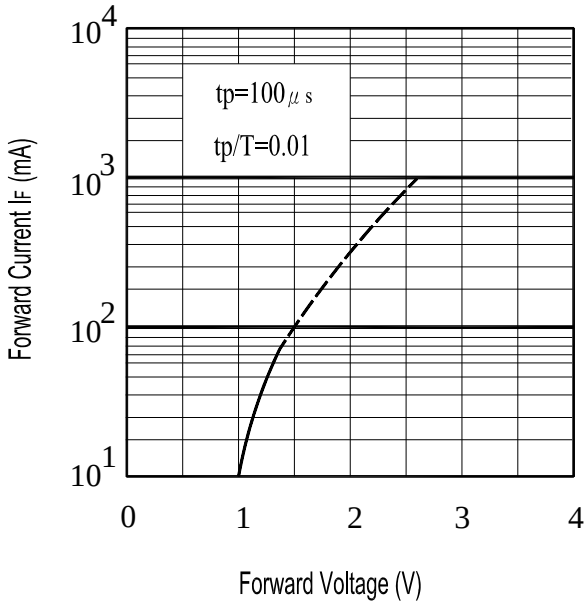


Fig.4 Forward Current vs. Forward Voltage



Typical Electro-Optical Characteristics Curves

Fig.5 Relative Intensity vs. Forward Current

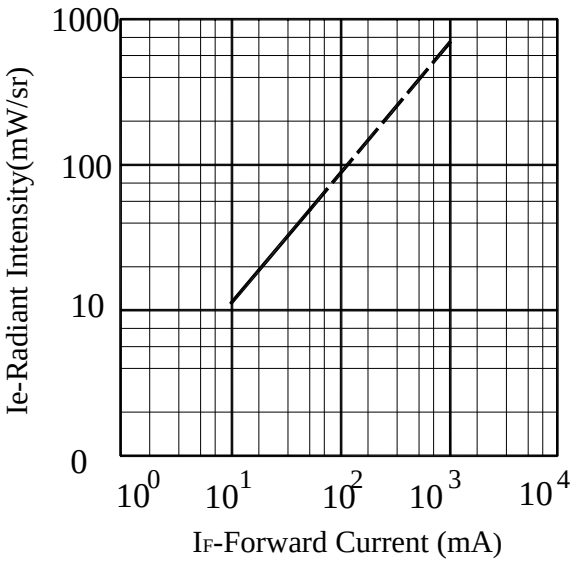


Fig.6 Relative Radiant Intensity vs. Angular Displacement

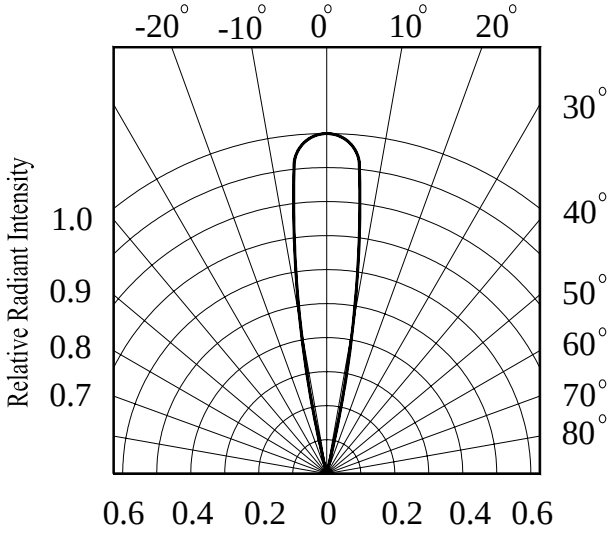


Fig.7 Relative Intensity vs. Ambient Temperature(°C)

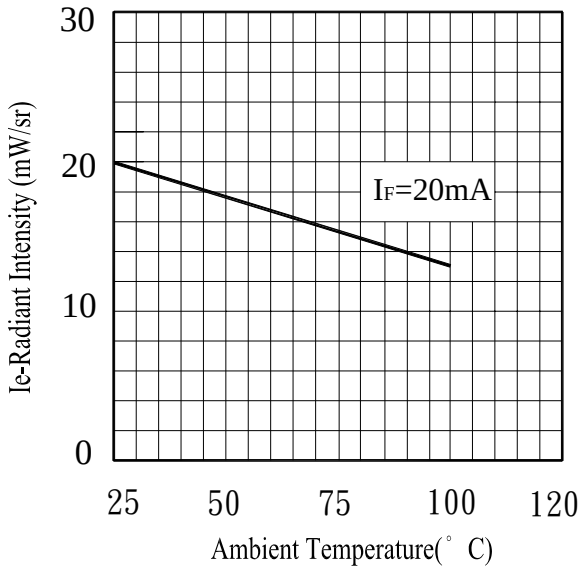
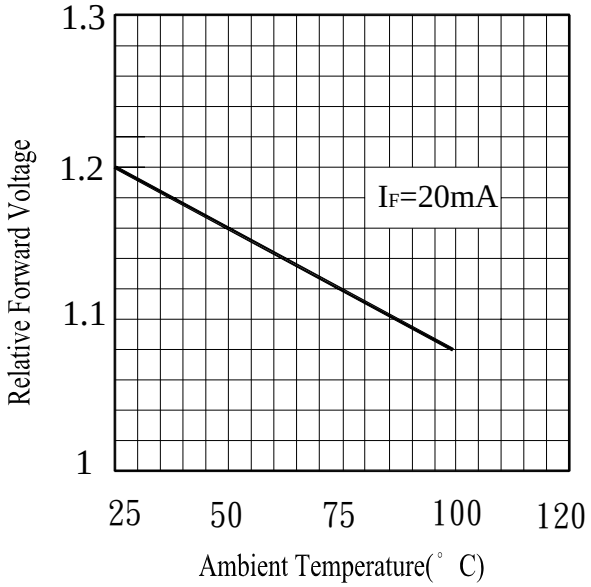


Fig.8 Forward Voltage vs. Ambient Temperature(°C)



Reliability Test Item And Condition

The reliability of products shall be satisfied with items listed below.

Confidence level：90%

LTPD：10%

NO.	Item	Test Conditions	Test Hours/ Cycles	Sample Sizes	Failure Judgement Criteria	Ac/Re
1	Solder Heat	TEMP.：260℃±5℃	10secs	22pcs	$I_R \geqq U \times 2$ $Ee \leqq L \times 0.8$ $V_F \geqq U \times 1.2$ U：Upper Specification Limit L：Lower Specification Limit	0/1
2	Temperature Cycle	H：+100℃ 15mins ↕ 5mins L：-40℃ 15mins	300Cycles	22pcs		0/1
3	Thermal Shock	H：+100℃ 5mins ↕ 10secs L：-10℃ 5mins	300Cycles	22pcs		0/1
4	High Temperature Storage	TEMP.：+100℃	1000hrs	22pcs		0/1
5	Low Temperature Storage	TEMP.：-40℃	1000hrs	22pcs		0/1
6	DC Operating Life	I_F=20mA	1000hrs	22pcs		0/1
7	High Temperature/ High Humidity	85℃ / 85% R.H	1000hrs	22pcs		0/1

Packing Quantity Specification

1.500PCS/1Bag，5Bags/1Box

2.10Boxes/1Carton

Label Form Specification

CPN: Customer's Production Number
P/N : Production Number
QTY: Packing Quantity
CAT: Ranks
HUE: Peak Wavelength
REF: Reference
LOT No: Lot Number
MADE IN TAIWAN: Production Place

Notes

1. Above specification may be changed without notice. EVERLIGHT will reserve authority on material change for above specification.
2. When using this product, please observe the absolute maximum ratings and the instructions for using outlined in these specification sheets. EVERLIGHT assumes no responsibility for any damage resulting from use of the product which does not comply with the absolute maximum ratings and the instructions included in these specification sheets.
3. These specification sheets include materials protected under copyright of EVERLIGHT corporation. Please don't reproduce or cause anyone to reproduce them without EVERLIGHT's

consent.

EVERLIGHT ELECTRONICS CO., LTD.
Office: No 25, Lane 76, Sec 3, Chung Yang Rd,
Tucheng, Taipei 236, Taiwan, R.O.C
Tel: 886-2-2267-2000, 2267-9936
Fax: 886-2267-6244, 2267-6189, 2267-6306
http:\\www.everlight.com

Silizium-PIN-Fotodiode
Silicon PIN Photodiode
Lead (Pb) Free Product - RoHS Compliant

SFH 213
SFH 213 FA

SFH 213

SFH 213 FA

Wesentliche Merkmale

- Wellenlängenbereich ($S_{10\%}$) 400nm bis 1100nm (SFH 213) und 750nm bis 1100nm (SFH 213 FA)
- Kurze Schaltzeit (typ. 5 ns)
- 5 mm-Plastikbauform im LED-Gehäuse

Features

- Wavelength range ($S_{10\%}$) 400 nm to 1100 nm (SFH 213) and 750nm to 1100nm (SFH 213 FA)
- Short switching time (typ. 5 ns)
- 5 mm LED plastic package

Anwendungen

- Industrieelektronik
- „Messen/Steuern/Regeln“
- Schnelle Lichtschranken

Applications

- Industrial electronics
- For control and drive circuits
- High speed photointerrupters

Typ **Type**	**Bestellnummer** **Ordering Code**	**Fotostrom**, E_V=1000 lx, standard light A, V_R = 5 V (SFH 213) **Photocurrent**, E_e=1 mW/cm^2, λ = 870nm, V_R = 5 V(SFH 213 FA) Ip (µA)
SFH 213	Q62702P0930	135 (≥100)
SFH 213 FA	Q62702P1671	90 (≥65)

OSRAM
Opto Semiconductors

Grenzwerte
Maximum Ratings

Bezeichnung Parameter	Symbol Symbol	Wert Value	Einheit Unit
Betriebs- und Lagertemperatur Operating and storage temperature range	T_{op}; T_{stg}	– 40 … + 100	°C
Sperrspannung Reverse voltage	V_R V_R (t < 2 min)	20 50	V V
Verlustleistung Total power dissipation	P_{tot}	150	mW

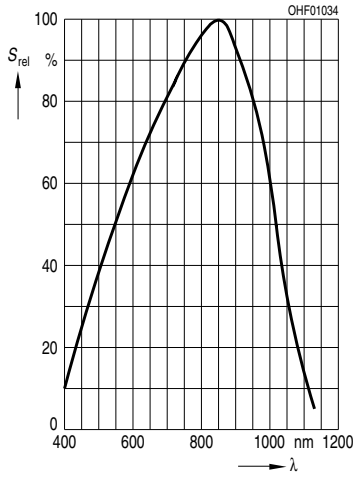
Kennwerte (T_A = 25 °C)
Characteristics

Bezeichnung Parameter	Symbol Symbol	Wert Value		Einheit Unit
		SFH 213	SFH 213 FA	
Fotostrom Photocurrent V_R = 5 V, Normlicht/standard light A, T = 2856 K, E_V = 1000 lx	I_P	135 (≥100)	–	µA
V_R = 5 V, λ = 870 nm, E_e = 1 mW/cm²	I_P	–	90 (≥65)	µA
Wellenlänge der max. Fotoempfindlichkeit Wavelength of max. sensitivity	$\lambda_{S\,max}$	850	900	nm
Spektraler Bereich der Fotoempfindlichkeit S = 10% von S_{max} Spectral range of sensitivity S = 10% of S_{max}	λ	400 …1100	750 … 1100	nm
Bestrahlungsempfindliche Fläche Radiant sensitive area	A	1	1	mm²
Abmessung der bestrahlungsempfindlichen Fläche Dimensions of radiant sensitive area	$L \times B$ $L \times W$	1 × 1	1 × 1	mm × mm
Halbwinkel Half angle	φ	± 10	± 10	Grad deg.
Dunkelstrom, V_R = 20 V Dark current	I_R	1 (≤ 5)	1 (≤ 5)	nA
Spektrale Fotoempfindlichkeit, λ = 870 nm Spectral sensitivity	S_λ	0.62	0.59	A/W
Quantenausbeute, λ = 870 nm Quantum yield	η	0.89	0.86	Electrons Photon

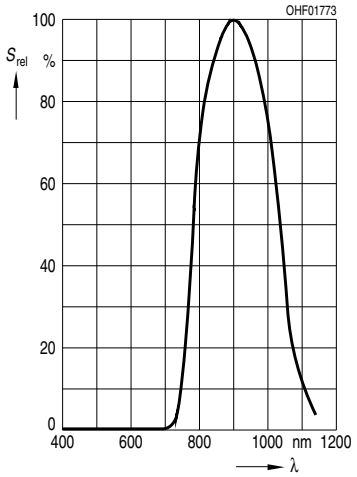
Kennwerte (T_A = 25 °C)
Characteristics (cont'd)

Bezeichnung **Parameter**	**Symbol** **Symbol**	**Wert** **Value**		**Einheit** **Unit**
		SFH 213	**SFH 213 FA**	
Leerlaufspannung Open-circuit voltage E_v = 1000 lx, Normlicht/standard light A, T = 2856 K E_e = 0.5 mW/cm², λ = 870 nm	V_O V_O	430 (≥ 350) –	– 380 (≥ 300)	mV mV
Kurzschlußstrom Short-circuit current E_v = 1000 lx, Normlicht/standard light A, T = 2856 K E_e = 0.5 mW/cm², λ = 870 nm	I_{SC} I_{SC}	125 –	– 42	µA µA
Anstiegs- und Abfallzeit des Fotostromes Rise and fall time of the photocurrent R_L = 50 Ω; V_R = 20 V; λ = 850 nm	t_r, t_f	5	5	ns
Durchlaßspannung, I_F = 80 mA, E = 0 Forward voltage	V_F	1.3	1.3	V
Kapazität, V_R = 0 V, f = 1 MHz, E = 0 Capacitance	C_0	11	11	pF
Temperaturkoeffizient von V_O Temperature coefficient of V_O	TC_V	– 2.6	– 2.6	mV/K
Temperaturkoeffizient von I_{SC} Temperature coefficient of I_{SC} Normlicht/standard light A λ = 870 nm	TC_I	 0.18 –	 – 0.1	%/K
Rauschäquivalente Strahlungsleistung Noise equivalent power V_R = 10 V, λ = 870 nm	NEP	2.9×10^{-14}	2.9×10^{-14}	$\frac{W}{\sqrt{Hz}}$
Nachweisgrenze, V_R = 20 V, λ = 870 nm Detection limit	D*	3.5×10^{12}	3.5×10^{12}	$\frac{cm \times \sqrt{Hz}}{W}$

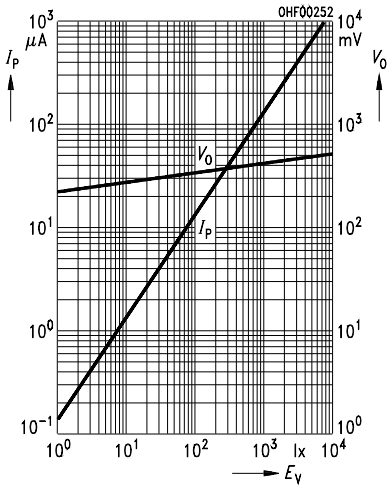
Relative Spectral Sensitivity
SFH 213, $S_{rel} = f(\lambda)$



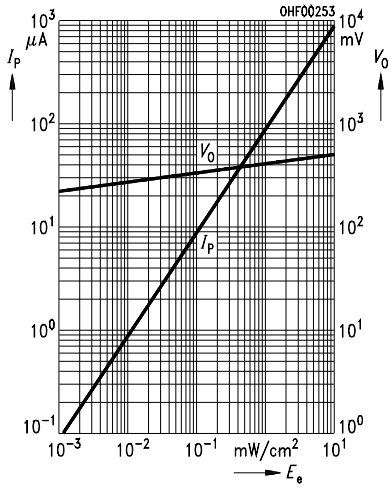
Relative Spectral Sensitivity
SFH 213 FA, $S_{rel} = f(\lambda)$



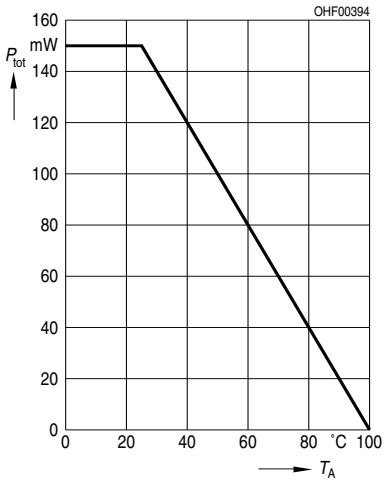
Photocurrent $I_P = f(E_v)$, V_R = 5 V
Open-Circuit Voltage $V_O = f(E_v)$
SFH 213



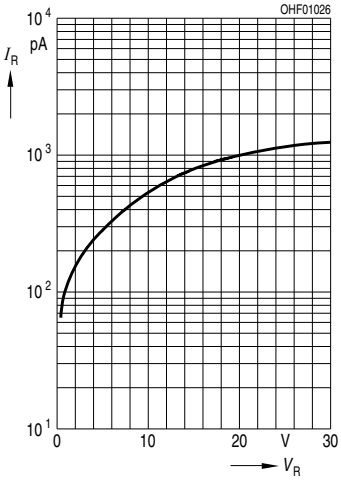
Photocurrent $I_P = f(E_e)$, V_R = 5 V
Open-Circuit Voltage $V_O = f(E_e)$
SFH 213 FA



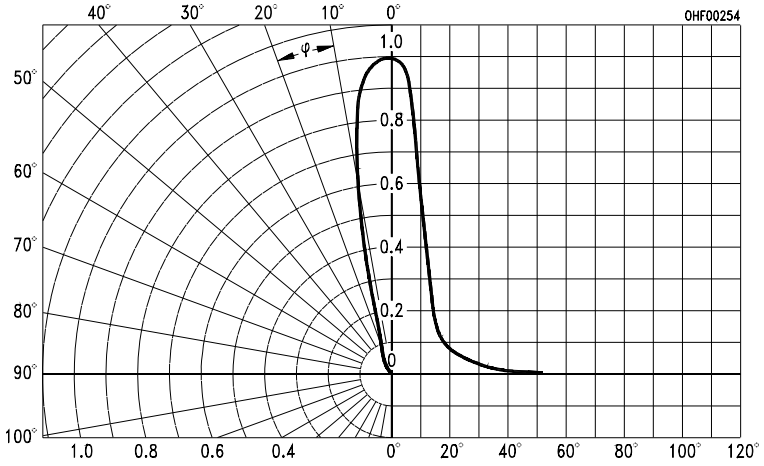
Total Power Dissipation
$P_{tot} = f(T_A)$



Dark Current
$I_R = f(V_R)$, E = 0

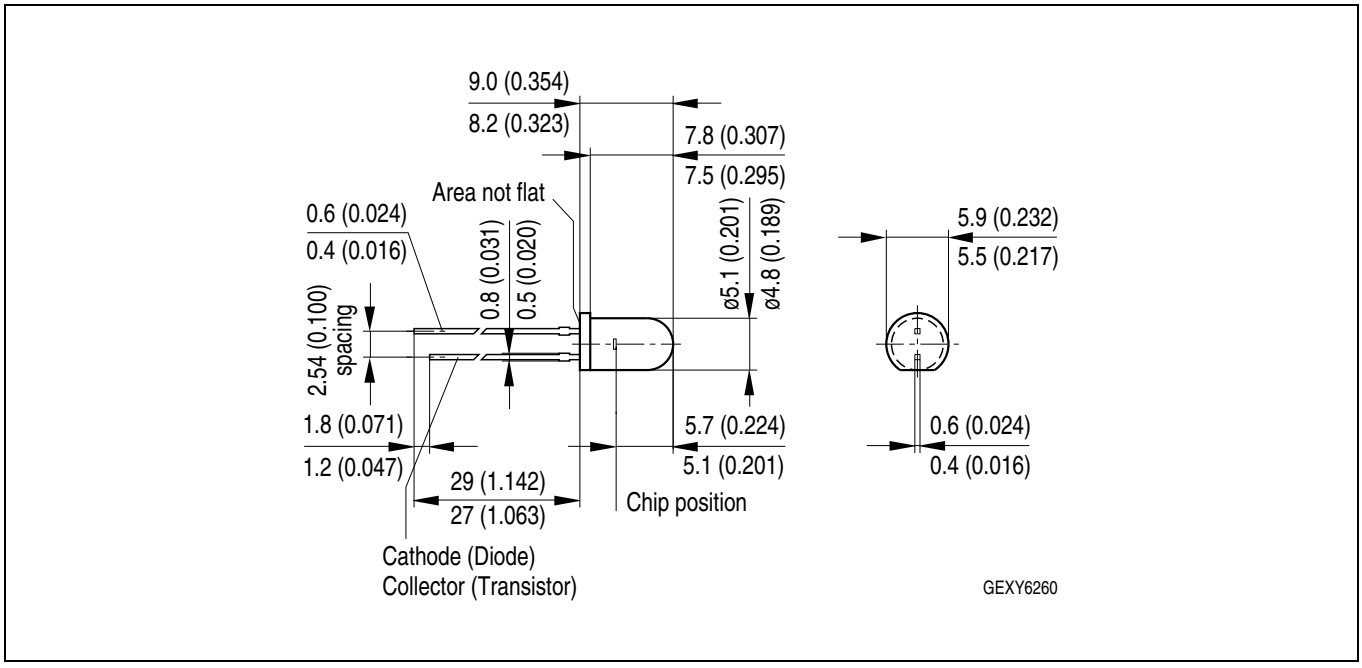


Directional Characteristics
$S_{rel} = f(\varphi)$



OSRAM
Opto Semiconductors

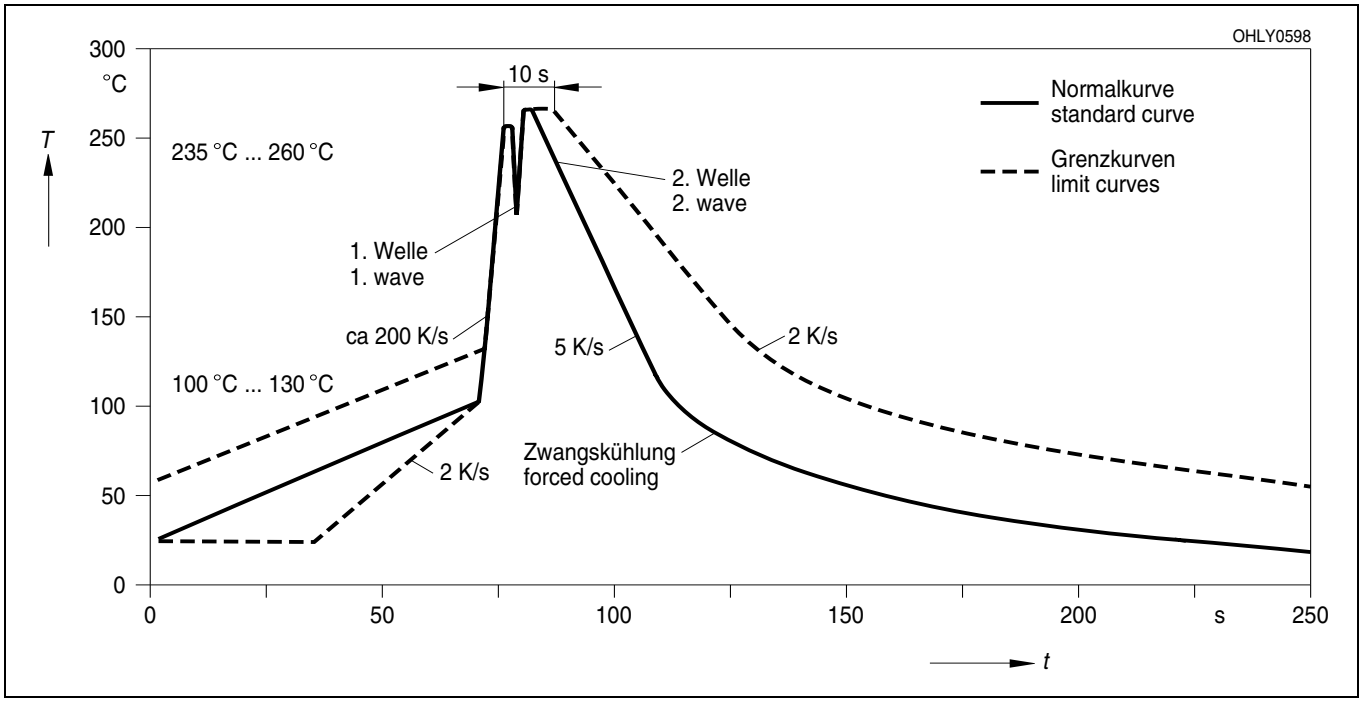
Maßzeichnung
Package Outlines



Maße in mm (inch) / Dimensions in mm (inch).

Lötbedingungen
Soldering Conditions
Wellenlöten (TTW)
TTW Soldering

(nach CECC 00802)
(acc. to CECC 00802)



OSRAM
Opto Semiconductors

EU RoHS and China RoHS compliant product

此产品符合欧盟 RoHS 指令的要求；

按照中国的相关法规和标准，不含有毒有害物质或元素。

Published by
OSRAM Opto Semiconductors GmbH
Leibnizstraße 4, D-93055 Regensburg
www.osram-os.com

The information describes the type of component and shall not be considered as assured characteristics.
Terms of delivery and rights to change design reserved. Due to technical requirements components may contain dangerous substances. For information on the types in question please contact our Sales Organization.

Packing

Please use the recycling operators known to you. We can also help you – get in touch with your nearest sales office. By agreement we will take packing material back, if it is sorted. You must bear the costs of transport. For packing material that is returned to us unsorted or which we are not obliged to accept, we shall have to invoice you for any costs incurred.

Components used in life-support devices or systems must be expressly authorized for such purpose! Critical components [1] , may only be used in life-support devices or systems [2] with the express written approval of OSRAM OS.

[1] A critical component is a component usedin a life-support device or system whose failure can reasonably be expected to cause the failure of that life-support device or system, or to affect its safety or effectiveness of that device or system.

[2] Life support devices or systems are intended (a) to be implanted in the human body, or (b) to support and/or maintain and sustain human life. If they fail, it is reasonable to assume that the health of the user may be endangered.