ELECTRÓNICA DIGITAL

EDITORIAL MACRO®

Perú - México - Colombia - Chile - Ecuador - España - Bolivia - Uruguay - Guatemala - Costa Rica

ELECTRÓNICA DIGITAL
Ingeniería de hardware
Autor: Gilberto González Rodríguez

Coordinación de edición:
Magaly Ramon Quiroz

Diseño de portada:
Alessandra Bonilla Zapata

Corrección de estilo:
Yossy Quintanilla Pinillos

Diagramación:
Eduardo Siesquén Aquije

Edición a cargo de:
© Empresa Editora Macro EIRL
Av. Paseo de la República N.° 5613, Miraflores, Lima, Perú

Teléfono: (511) 748 0560
E-mail: proyectoeditorial@editorialmacro.com
Página web: www.editorialmacro.com

Primera edición: marzo 2016

ISBN N.° 978-612-304-344-5
ISBN e-book N.° 978-612-304-371-1

Gilberto González Rodríguez

Nació en México. Ingeniero en Sistemas Computacionales y magíster en Tecnologías de la Información y la Comunicación (TIC), con una especialidad en el desarrollo de sistemas de computación y comunicaciones.

Ha participado como ponente, profesor, desarrollador y asesor en diversas instituciones y congresos. Asimismo, en cursos como diseño de hardware y prototipos a escala, robótica aplicada, dictado por la UNAM, y desarrollo de proyectos con Arduino (impartido por la IEEE y el IPN).

Entre sus publicaciones se encuentran *Servicio técnico: Notebook* y *Configuración de routers y switches CISCO*. Colaboró en el material *Técnico en redes y seguridad* como autor del e-book *Solución de problemas de redes*.

EDITORIAL
MACRO®

Dedicatoria

Este libro está dedicado con mucho cariño a ese ser tan especial que hoy forma parte de mi vida. Aquel que ha sido mi fuente de inspiración durante todo este tiempo, quien con su sonrisa ilumina mi mundo y le da sentido a mis días. A mi hijo Rodrigo Gilberto G. E.

EDITORIAL
MACRO®

Agradecimientos

A mi editora Magaly Ramon por su tiempo, entrega y constante dedicación durante la elaboración de este material.

A Editorial Macro y a todos los implicados en la producción editorial de este libro, sin su confianza este sueño no sería posible.

EDITORIAL
MACRO®

Índice

Capítulo 7: Prácticas con Arduino

Capítulo 8: Microcontroladores

Apéndice

INTRODUCCIÓN

El presente libro refleja un logro más en el quehacer profesional del maestro Gilberto González Rodríguez, quien tiene el gusto de compartir en esta ocasión sus conocimientos sobre temas relacionados con la ingeniería de hardware, la electrónica digital, el desarrollo de proyectos y la robótica. Sin duda, se trata de una obra que no puede faltar en la biblioteca personal de un aficionado a la electrónica, o de todo aquel estudiante de carreras afines. Este libro contiene las bases y técnicas necesarias para garantizar al lector un aprendizaje simple a través de ejemplos, ejercicios y prácticas.

En un inicio se ofrece una introducción al libro, en la que se describen los conceptos básicos y se proporcionan las bases que le harán pensar al usuario en la posibilidad de comenzar a desarrollar un proyecto funcional en el futuro. Durante el recorrido se describen los sistemas numéricos para efectuar conversiones entre distintas bases, los cuales son aplicados más adelante en la generación de circuitos lógicos, tema que ofrece un contexto ameno, pero sobre todo muy interesante en cuanto a la construcción de circuitos bajo simulación y de manera física, lo que implica el conocimiento de componentes electrónicos y herramientas de montaje para efectuarlos.

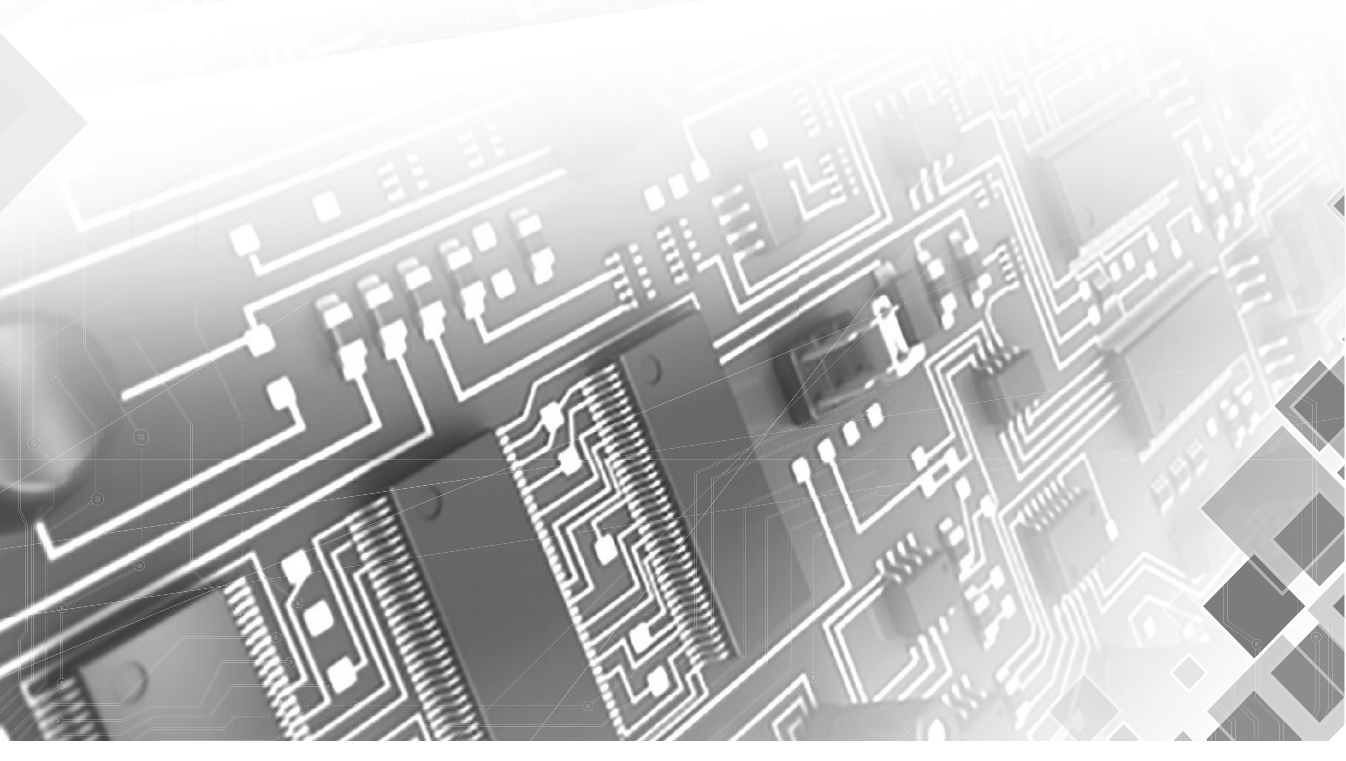
Luego, se da a conocer los diferentes sistemas digitales que permiten mostrar un panorama general de funcionamiento sobre proyectos de lógica compleja, pasando por el uso, programación y aplicación de microcontroladores, tema en el que se analiza la plataforma Arduino, su manejo y programación. Además, se explica una forma sencilla de programar un microcontrolador a través de diagramas de flujo. Complementariamente, se ofrece una descripción general de las herramientas de software auxiliares para el análisis, simulación, desarrollo e implementación de proyectos de electrónica digital e ingeniería de hardware, citando a su vez algunos métodos, herramientas y procedimientos para la generación de productos con el uso de placas de circuito impreso (PCB).

Como se puede apreciar en dicho contenido, se retoman temas imprescindibles, sin dejar de considerar el planteamiento de múltiples actividades, complementadas con diez prácticas de laboratorio.

EDITORIAL
MACRO®

Capítulo 1

Introducción a la ingeniería de hardware



EDITORIAL
MACRO®

1.1 CONCEPTOS BÁSICOS DE LA ELECTRÓNICA

Antes de entrar al tema de electrónica digital, es necesario comenzar a definir algunos términos básicos, de los que se hablará durante todo el contexto del presente libro. Términos como: electrónica, señal, circuito digital, sistema, componente electrónico, dispositivo electrónico, hardware, puertos e interfaces, lenguaje de programación. A continuación se describen e ilustran cada uno de estos:

a. Electrónica: Este concepto es habitualmente asociado o a veces confundido por otros términos de similar contexto, como electricidad, robótica, eléctrica e incluso mecatrónica. No obstante, aunque mantienen una estrecha relación, son conceptos totalmente diferentes. Electrónica se define, sencillamente, como parte de la física (y, desde luego, de la ingeniería) que hace estudio de las señales como medio para transmitir información. En otras palabras, se trata de un estudio tanto de los impulsos eléctricos como de la escala de los posibles valores inmersos en una señal. Como se puede apreciar, la definición del término «electrónica» cede lugar a un concepto más de suma importancia, señal.

b. Señal: Se define como una forma de dato, la cual es usualmente tratada como una secuencia de valores de una escala cuantitativa (amplitud) registrada (interpretada, tabulada, graficada) contra el tiempo. Las señales pueden ser fácilmente visualizadas gracias al uso de un osciloscopio (aparato capaz de mostrar en pantalla la oscilación de ondas que genera una señal), lo que permite un estudio completo de su comportamiento. Las señales pueden ser tratadas como magnitud, ya que expresan cantidades cuantificables.

Se debe saber que para el análisis de una señal, entre otros aspectos, han de considerarse varios parámetros como la tensión (V), la frecuencia (F), el periodo de tiempo (T), la amplitud (A), etc. Los cuales serán analizados más adelante en el capítulo siete (Prácticas con Arduino) de este libro.

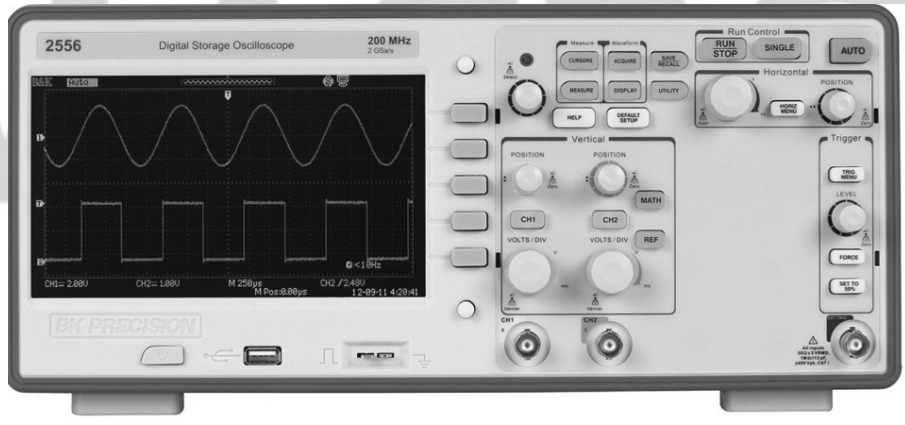


Figura 1.1 El osciloscopio, herramienta útil para el estudio de las señales

c. Circuito digital: Hace referencia al tipo de circuito capaz de representar una señal (como entidad compuesta por una secuencia de valores), que solo maneja dos posibles estados o niveles de tensión: HIGH y LOW. Los circuitos digitales, a diferencia de los circuitos analógicos, poseen una mayor escala de integración en la vida real. Son mucho más exactos, fáciles de interpretar y por lo mismo más confiables. Los circuitos digitales, por lo general, parten de la lógica binaria para representar cantidades en decimal u otro sistema numérico. Gracias a este tipo de circuitos, se pueden construir contadores, relojes, alarmas, cajas de seguridad, termómetros digitales, cronómetros, sistemas de iluminación, riego e incluso computadoras.

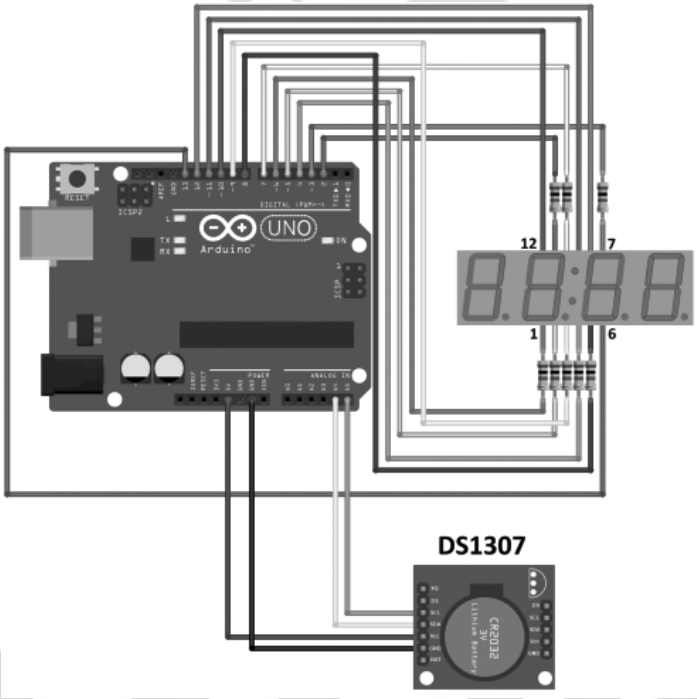


Figura 1.2 Un reloj digital como ejemplo de un circuito digital

d. Sistema: Es definido como un conjunto de componentes que interactúan entre sí para lograr un fin común. Y, aunque el enfoque de principio no es de índole electrónico, este concepto es catalogado como de suma importancia en el estudio de asignaturas como la electrónica digital e ingeniería de hardware. Este término va de la mano con el concepto de señal y electrónica.

Actualmente, existe una gran variedad de campos de aplicación (aparte de la electrónica) en el que se encuentra presente la concepción de señales y sistemas. Entre estos el sector de las comunicaciones, el rubro de los sistemas de control, la domótica (ardumótica, término recientemente empleado para hacer referencia a la automatización a través del sistema Arduino), nanotecnología, mecatrónica/robótica, entre otros. Más adelante, se conocerán algunos ejemplos de aplicación real de este concepto.

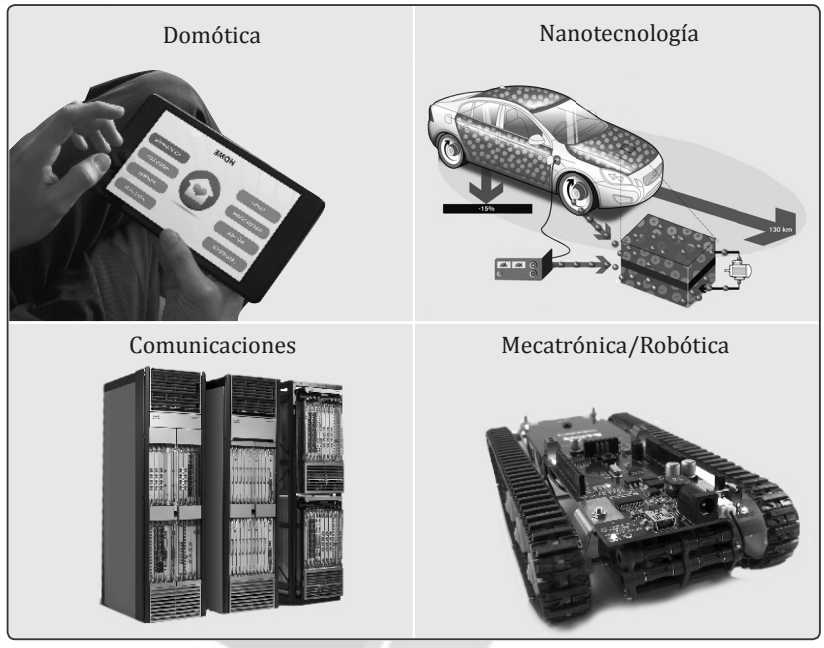


Figura 1.3 Campos de aplicación de la electrónica en la vida real

e. Componente electrónico: Es aquella pieza integral que forma parte de un sistema electrónico, el cual es capaz de comportarse de acuerdo a la función establecida por un circuito. El componente electrónico tiene como propósito afectar (de manera programada) tanto el campo como el flujo de electrones presentes en un sistema.

Existe una gran variedad de componentes electrónicos, estos a menudo se clasifican de acuerdo a su función, grado de acción o aplicación. Algunos ejemplos de componentes electrónicos son: diodos, relevadores, interruptores, circuitos integrados, resistencias, capacitores, condensadores y transistores.

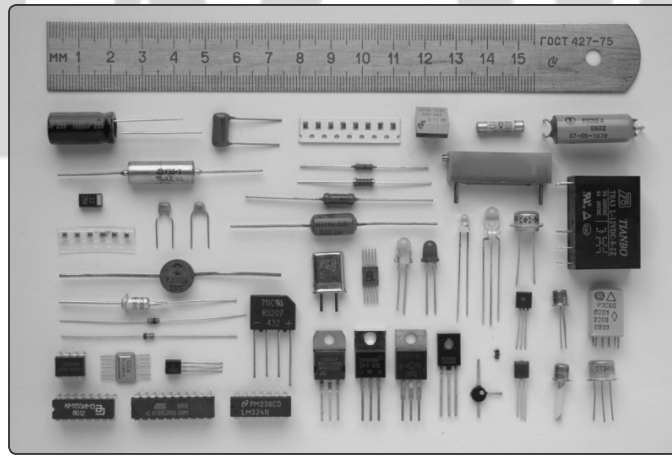


Figura 1.4 Algunos componentes electrónicos utilizados en la electrónica digital

f. Dispositivo electrónico: Se trata de un sistema o módulo integrado por un conjunto de componentes electrónicos que cumple con un propósito en particular. Por lo general, un dispositivo electrónico se encuentra conformado por una placa de circuito impreso (*print circuit board* de sus siglas en inglés o simplemente PCB) donde se alojan componentes que respaldan su función.

El término «dispositivo electrónico» puede tener dos enfoques, según su aplicación, puede tratarse como equipo electrónico (generalmente de marca), pero también puede tratarse de un elemento auxiliar para el desarrollo e integración de proyectos.

Algunos ejemplos de dispositivos electrónicos son cualquier periférico para PC (*personal computer*), un amplificador de audio, una pantalla LCD, una placa Arduino (la cual será objeto de estudio más adelante), un router o switch.

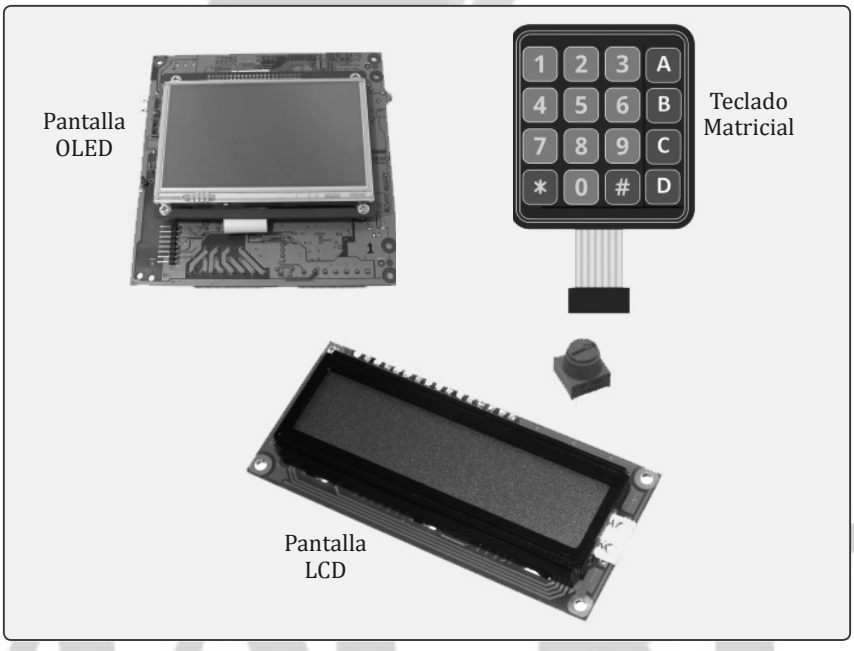


Figura 1.5 La pantalla LCD y el teclado matricial, ejemplos de dispositivos electrónicos

g. Hardware: En términos informáticos, se define como el conjunto de elementos tangibles a diferencia del software. El hardware se conforma por módulos y, estos a su vez, por componentes electrónicos (por lo general, son digitales). El hardware de una computadora, por ejemplo, se compone de, entre otros elementos, por puertos e interfaces de comunicación. Lo anterior ha hecho posible la interacción con el usuario u operador. El software representa, sin duda, unos elementos imprescindibles para el funcionamiento del hardware y viceversa.

h. Puertos/interfaces: Se trata de elementos físicos integrados sobre PCB funcionales. Un puerto se define como receptáculo para la colocación de dispositivos electrónicos extraíbles y periféricos, en tanto que las interfaces evocan el tipo de tecnología adoptada para su conexión. El módulo de *On-board* de una computadora es un claro ejemplo de puertos e interfaces electrónicas. Otro ejemplo de integración de puertos e interfaces es la plataforma Raspberry Pi.

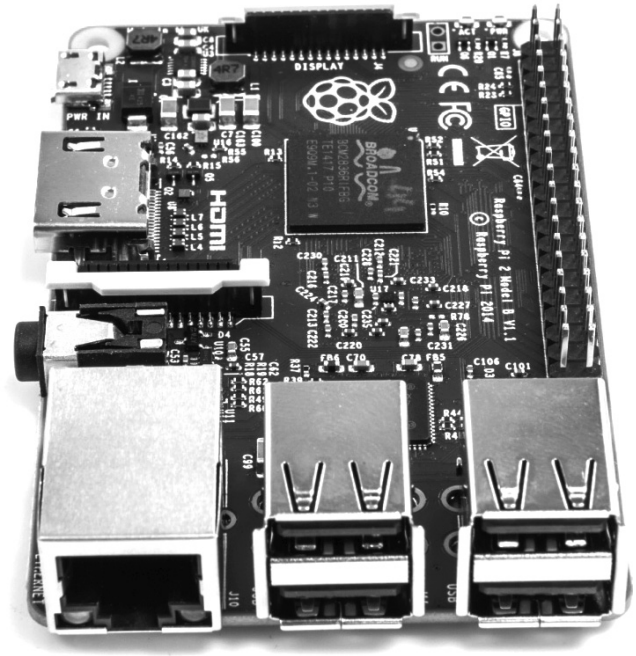


Figura 1.6 El uso de puertos e interfaces se encuentra presente en computadoras como la Raspberry Pi

i. Lenguaje de programación: Se trata de un entorno que cuenta con un sistema de interpretación, compilación, ejecución y depuración de código fuente (el cual cuenta con una serie de instrucciones que ejecutan cualquier tarea). Los lenguajes de programación, a menudo, son utilizados para conseguir la comunicación entre una PC y el mismo usuario. A lo largo del tiempo, la electrónica digital ha hecho posible la integración de lenguajes de programación y entornos de desarrollo para la programación, no solo de componentes electrónicos, sino también de dispositivos. No cabe duda que la convivencia entre el campo de la informática y el campo de la electrónica han dado lugar a nuevos paradigmas con los cuales se puede ofrecer algún tipo de respuesta a las demandantes necesidades de la sociedad.

Algunos ejemplos de lenguajes de programación son: lenguaje C (C++), Java, HTML, Python, entre otros. Más adelante, en este mismo capítulo, se profundizará sobre la programación de hardware y, en especial, del lenguaje C.

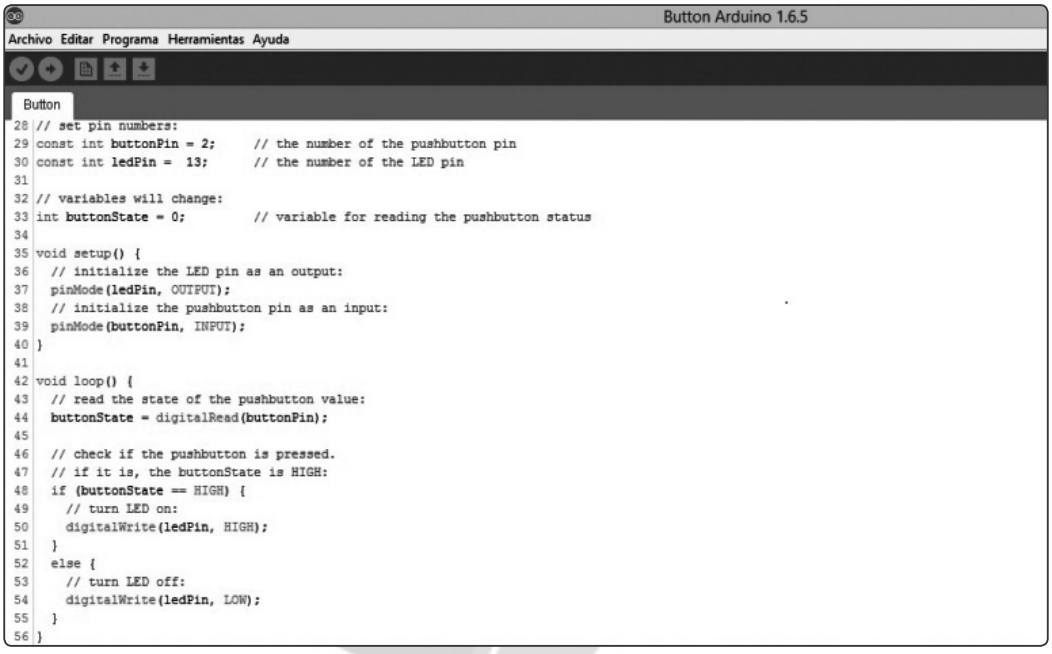


Figura 1.7 Lenguaje C, para programar interfaces electrónicas

ACTIVIDAD 1

1. Defina, con sus propias palabras, el término «electrónica».
2. Mencione por lo menos tres campos de aplicación de la electrónica.
3. Investigue y liste los componentes electrónicos necesarios para la construcción de un robot diferencial.
4. Mencione la diferencia entre el término «puerto» y el término «interfaz».
5. Investigue y mencione por lo menos tres proyectos que se puedan realizar con el uso de la plataforma Raspberry Pi.
6. Investigue las estructuras de control utilizadas en un lenguaje de programación.

1.1.1 Electrónica y señales

Hasta el momento se ha definido tanto el término «electrónica» como el término «señal», por lo tanto, solo resta profundizar respecto a características, representación, ventajas y desventajas de los tipos de señales (magnitudes) existentes en la electrónica, la **señal analógica** y la **señal digital**, siendo esta última el objeto de estudio del presente material. A menudo, ambos tipos de señales o representaciones pueden combinarse, pues en el sector industrial y en la vida cotidiana suelen aplicarse para la resolución de problemas. Algunos ejemplos de combinación de señales analógicas y digitales se encuentran presentes en:

- Sistemas de reproducción de CD de música: Desde un inicio, la información contenida en un CD musical se encuentra digitalizada, una vez que la unidad reproductora inicia su lectura, transforma la magnitud digital origen a una magnitud analógica. Lo anterior se consigue gracias a un convertidor de señal digital a analógica (DAC, *digital to analog converter*). El resultado obtenido consiste en un conjunto de ondas sonoras emitidas por un sistema que funciona como dispositivo de salida de audio.

  Es necesario recordar que, en términos informáticos (en específico si se habla de una computadora), el lector o reproductor se conoce como dispositivo de entrada o lectura de datos, en tanto que cualquier dispositivo por donde sale la información, que en este caso puede ser una bocina, un *buffer* o un audífono, se llama dispositivo de salida.

- Sistemas de grabación o digitalización de audio: Se trata del proceso inverso al punto antes mencionado. Un ejemplo real son los estudios de grabación, en los cuales, para grabar un CD (película o audio) se utiliza un convertidor de magnitudes analógicas a digitales, comúnmente llamado ADC (*analog to digital converter*).

- Sistema de televisión: Se sabe que la transmisión de televisión se puede efectuar de manera analógica o digital. Para conseguir esa transformación se emplean dispositivos ideales conectados a una antena aérea, la cual se comunica vía satélite. Estas antenas pueden ser externas o internas.

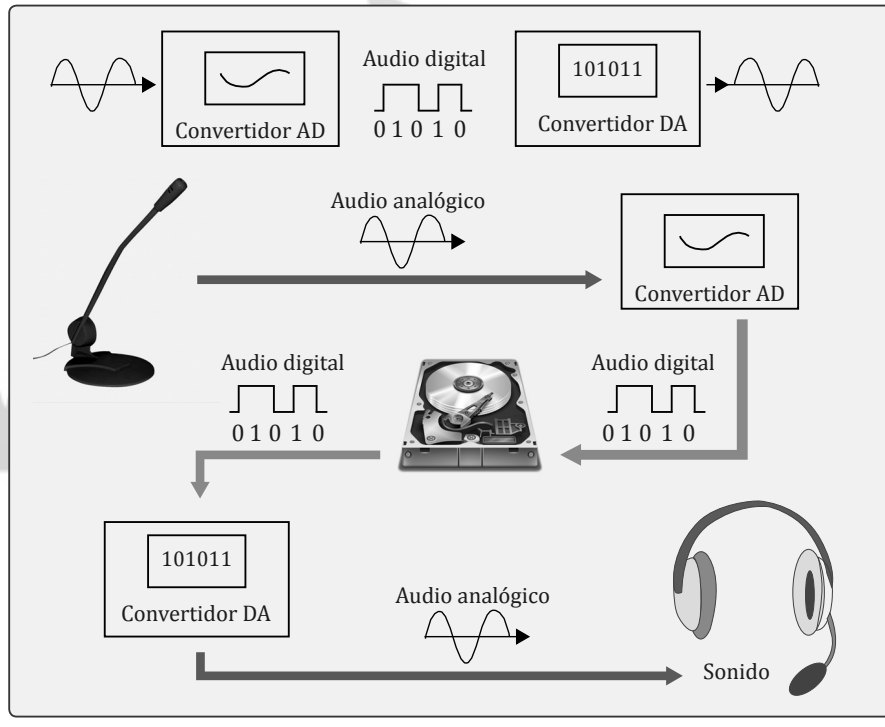


Figura 1.8 Un estudio de grabación, un claro ejemplo de combinación de señales analógicas y digitales
Fuente: el autor.

A. Representación analógica y digital

El término «analógico» se refiere a las magnitudes o valores que varían con el tiempo en forma continua. Generalmente, estas magnitudes se encuentran presentes en el medio ambiente. Algunos ejemplos son: la distancia, la temperatura, la velocidad, la presión, etc.

A continuación se listan algunas características de las magnitudes analógicas:

- Cuentan con un rango infinito de valores.
- Muestran valores aproximados, nunca exactos.
- La representación gráfica de una magnitud analógica es mediante una onda senoidal.
- Presentan una escala de valores continuos.
- Se ven afectadas por el ruido y los factores externos como campos magnéticos o condiciones ambientales.

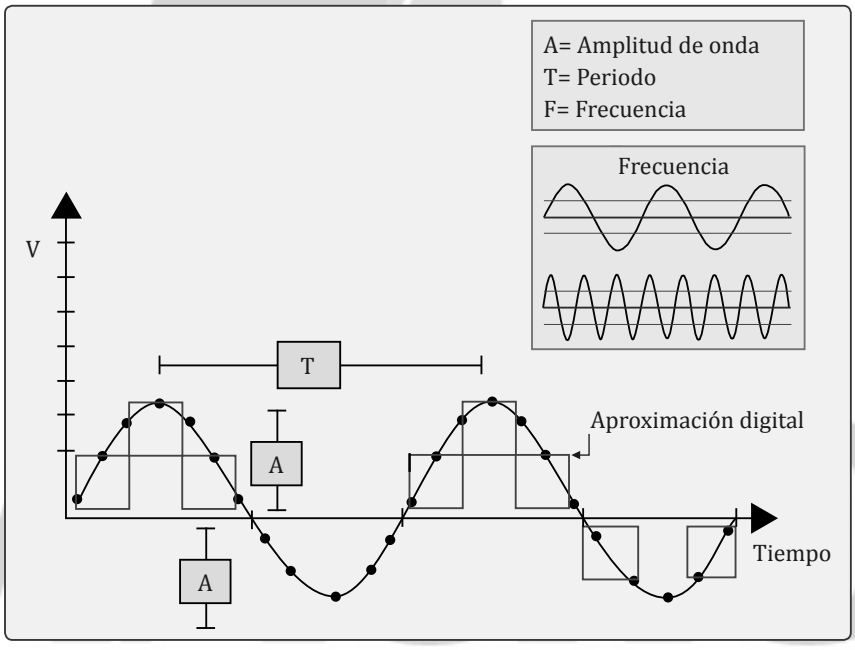


Figura 1.9 Esquema de representación de una magnitud analógica
Fuente: el autor.

La señal digital representa las magnitudes o valores registrados contra el tiempo en forma discreta a diferencia de la magnitud analógica. La palabra «digital» tiene origen latino *digitus*, 'contar con los dedos'. Actualmente, la mayoría de los dispositivos o elementos analógicos han evolucionado, dando lugar al diseño e implementación de equipos digitales. La representación digital, por lo general, se encuentra compuesta en forma de ondas con un pulso positivo y un pulso negativo, los cuales forman una señal cuadrada.

A continuación se listan algunas características de las magnitudes digitales:

- La representación gráfica de esta magnitud es mediante el uso de dos posibles valores (0 y 1).
- Se encargan de mostrar valores exactos.
- La representación gráfica de una magnitud analógica es mediante una señal cuadrada (escalón por escalón).
- Presentan una escala de valores discretos.
- Son inmunes al ruido, factores externos o condiciones ambientales.

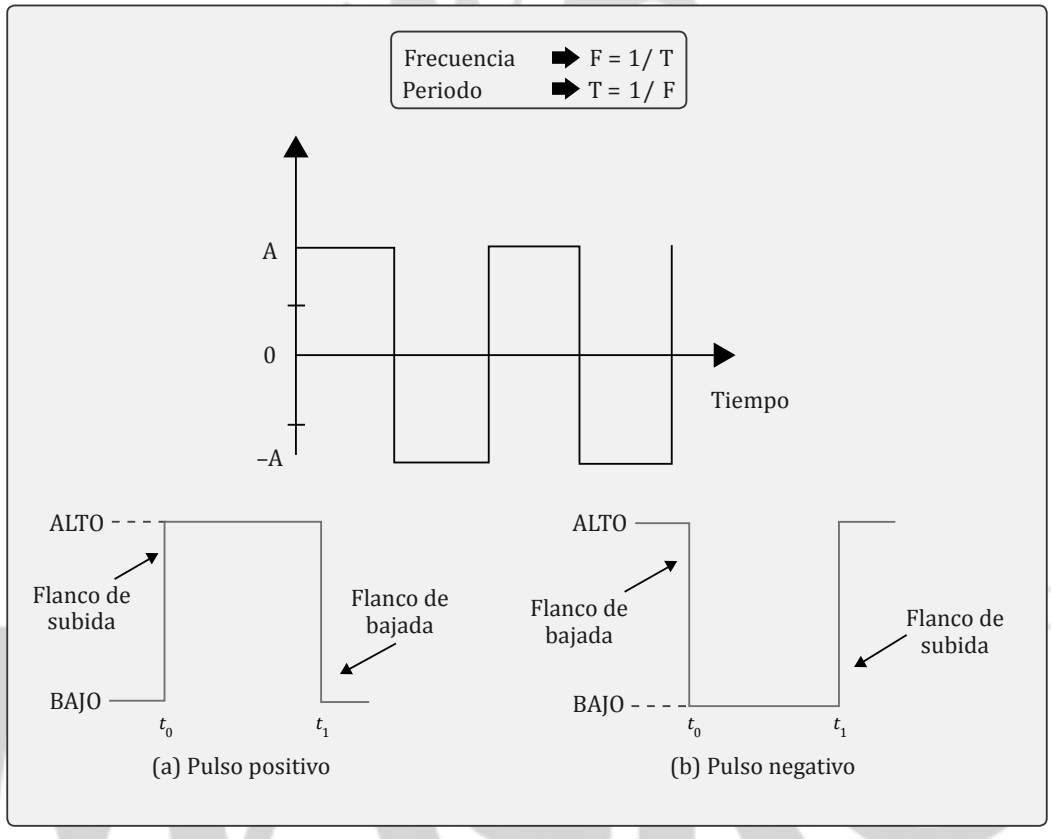


Figura 1.10 Esquema de representación de una magnitud digital
Fuente: el autor.

ACTIVIDAD 2

Dados los siguientes ejemplos, determine cuáles implican el manejo de valores analógicos y valores digitales.

a) Interruptor de 8 posiciones (DIP *switch*): _____

b) Corriente que fluye fuera de una toma de corriente (CA): _____

c) Temperatura de un espacio: _____

d) Velocímetro de un automóvil: _____

e) Toma turnos de una sucursal bancaria: _____

f) Reloj de arena: _____

B. Niveles de tensión y familias lógicas

En el campo de la electrónica digital existen dos niveles de tensión, el nivel alto (HIGH), el cual se representa con 1 lógico y; el nivel de tensión bajo, con 0 lógico. Ambos niveles tienen asociado un voltaje, 5 V para el nivel HIGH y 0 V para el nivel LOW. En la tabla 1.1 se puede apreciar el valor lógico, el símbolo, el valor de tensión y los efectos de realización que cumplen ambos niveles de tensión.

Tabla 1.1 *Principales características de los niveles de tensión presentes en la electrónica digital*

Valor lógico	1 lógico	0 lógico
Símbolo (interruptor)	___./ .___	___.___.___
Descripción	Nivel de tensión alto (HIGH)	Nivel de tensión bajo (LOW)
Voltaje	5 V	0 V
Efectos de funcionamiento	Existe un paso de corriente	No existe paso de corriente

La razón principal, por la cual se han adoptado estos dos niveles de tensión en muchos dispositivos electrónicos, se debe a que representan valores más fáciles de distinguir, interpretar, incluso implementar. Por lo tanto, el coste de un dispositivo electrónico es relativamente barato.

Como se ha mencionado, los niveles de tensión equivalen a un nivel de voltaje, el cual puede incluso variar, de acuerdo a la tecnología empleada. Las tecnologías más comunes para la construcción de circuitos (con el uso de circuitos integrados) son la tecnología CMOS (*complementary metal oxide semiconductor*, semiconductor de óxido de metal complementario) y la tecnología TTL (*transistor transistor logic*, lógica de transistor), la cual será objeto de estudio en este material. Ambas son definidas como familias lógicas y a menudo, se encuentran clasificadas en series.

De acuerdo a lo anterior, es importante tener en cuenta que, al trabajar con circuitos lógicos, se deben considerar algunos parámetros, sobre todo si se trata de voltaje, potencia (HIGH o LOW), velocidad. Desde luego, que estos parámetros están dados según la familia que se desea utilizar.

A menudo, cuando se trabaja con circuitos integrados, es común encontrar datos como DIP-CI-SN74LS08N, o DIP-CI-SN74ALS32N, los cuales son fáciles de leer e interpretar. Para ello, es necesario recurrir al análisis de la serie del dispositivo. Algunas de ellas se muestran a continuación:

Tabla 1.2 *Series de dispositivos TTL*

Serie	Descripción
74	TTL estándar
74H	TTL de alta velocidad
74S	TTL Schottky
74LS	TTL Schottky de baja potencia
74AS	TTL Schottky avanzado
74ALS	TTL Schottky avanzado de baja potencia
74F	TTL Fast (rápido)

En el capítulo tres, se profundiza la explicación sobre las series más comunes en circuitos integrados y sus inscripciones.

1.1.2 Componentes y dispositivos electrónicos

Con anterioridad se ha explicado la diferencia existente entre la definición de componente y dispositivo electrónico. Asimismo, se ha hecho mención de algunos ejemplos de cada uno. En el presente tema, se desarrolla la descripción detallada tanto de algunos componentes como dispositivos electrónicos utilizados en el ámbito de la ingeniería de hardware.

A. Los componentes electrónicos

Actualmente, existen diferentes fabricantes tanto de componentes electrónicos como de dispositivos, dando lugar, a su vez, a una gran variedad de estos elementos en el mercado. Dado esto, es necesario comenzar a clasificarlos y ordenarlos con el único fin de reconocerlos y comenzar a utilizarlos, de acuerdo a la necesidad del usuario.

La clasificación de los componentes electrónicos se muestra en la siguiente tabla:

Tabla 1.3 *Clasificación de los componentes electrónicos*

Clasificación	Descripción	Componente
Fuentes de alimentación	Incluye fuentes reguladas que permiten polarizar (VCC y GND) cualquier circuito para su funcionamiento	Baterías de 1.5, 3,3.3 y 5 y 9, 12 V, fusibles, reguladores de voltaje, fuentes DC
Conectores	Estos se agrupan según su uso y aplicación. Son elementos pasivos para la conexión de nodos, los cuales conducen voltaje o datos	Zócalos o sockets, bloques, SIL y DIL
Single-In-Line (SIL-Header)	Son elementos pasivos que permiten la conductividad de corriente o voltaje en una sola línea. Se trata de un conjunto de terminales (hembra o macho) conocidos como *headers*	*Headers* hembra, macho de una sola línea (1-25 pines), cables jumper
Dual-In-Line (DIL-Header)	Son elementos como el SIL, solo que estos permiten la conductividad de corriente o voltaje en dos líneas	*Headers* hembra, macho de dos líneas (de 4 a 40 pines)
Bloques/ Terminales	Incluye elementos pasivos que permiten la conexión de un sistema de polarización (VCC y GND)	Terminales de conexión VCC y GND (2-4 CONTACTOS)
Zócalos	Son receptáculos para la colocación de chips. Son elementos pasivos de conductividad y ajuste	Porta chips de 4 a 50 pines
Componentes de entrada	Se llama así a todos los componentes capaces de dirigir un pulso al circuito electrónico utilizado	*Switches* o interruptores de distintos tipos, sensores de distintos tipos e instrumentos de virtualización (generadores de señal)
Componentes pasivos	Son elementos pasivos que sirven como conductores (elementos que se oponen al paso de la corriente o se encarga de almacenar energía)	Resistencias, capacitores, inductores (transformadores), potenciómetros
Semiconductores	Son elementos que se comportan como conductores de la corriente o aislante	Diodos y transistores (NPN, PNP)
Circuitos integrados	Componentes o chips que integran diversas funciones. Son de propósito específico, pueden ser programables y no programables	Compuertas lógicas, *timers*, *flipflops*, amplificadores, sumadores, multiplexores, codificadores y decodificadores, comparadores, etc.
Componentes de salida	Son los componentes capaces de mostrar un resultado esperado con base a la emisión y procesamiento de pulsos	Bulbos, LED, *buzzers*, *speakers*, *display* (7, 16 segmentos, duales, matrices), elementos electromagnéticos (motores, relevadores o relés)

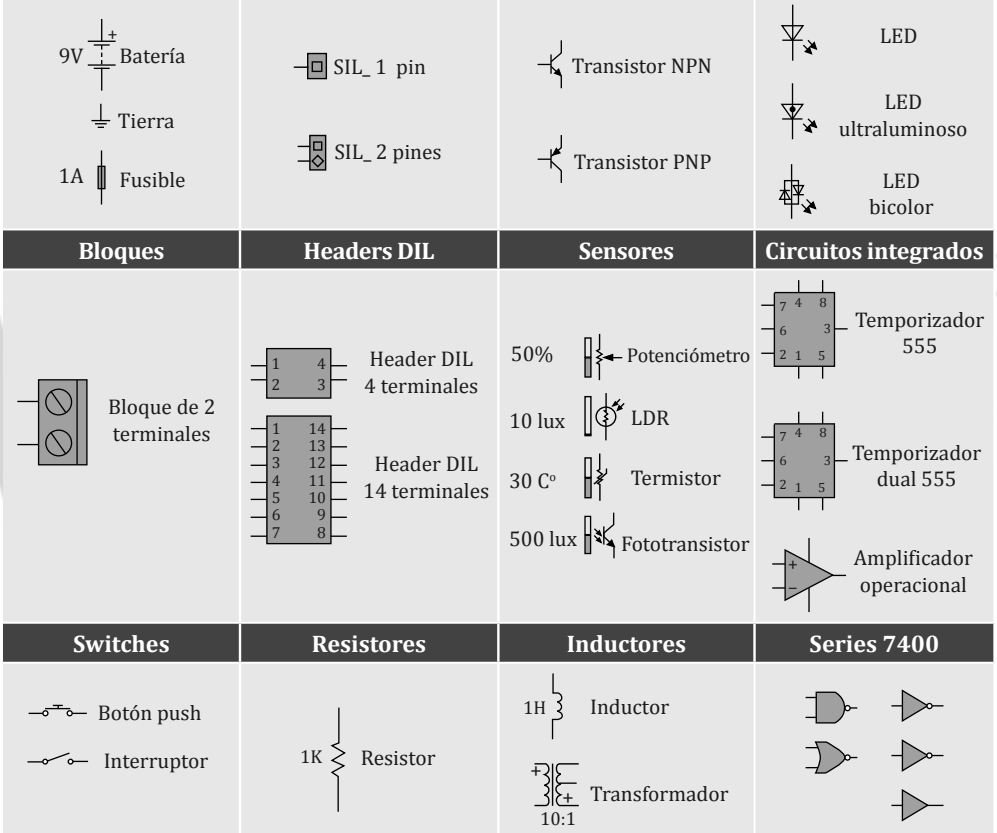
Un par de dispositivos electrónicos más, que hay que tratar de manera específica, es el **microprocesador** (CPU) y el **microcontrolador**. Los cuales se describen más adelante en este capítulo.

Si en algún momento se precisa conocer más a fondo estos elementos, se puede recurrir a la instalación y uso de la herramienta PCB *wizard*/*liveWire*. Utilería de software, generalmente, utilizada para montar circuitos y mostrarlos sobre placas de circuito impreso o PCB (*print circuit board*). Este software muestra en uno de sus paneles el símbolo o diagrama estándar de los componentes electrónicos más comunes.

Los componentes electrónicos, a menudo, son asociados o representados con un símbolo (simbología americana y europea); esto resulta útil al momento de diseñar algún esquema de conexión (diagramas lógicos, topológicos o de contacto).

En la siguiente imagen se ilustran los componentes electrónicos tanto analógicos como digitales más representativos y su respectivo símbolo.

Fuentes de alimentación	Headers SIL	Transistores	Diodos emisores de luz
9V Batería Tierra 1A Fusible	SIL_ 1 pin SIL_ 2 pines	Transistor NPN Transistor PNP	LED LED ultraluminoso LED bicolor
Bloques	**Headers DIL**	**Sensores**	**Circuitos integrados**
Bloque de 2 terminales	Header DIL 4 terminales Header DIL 14 terminales	50% Potenciómetro 10 lux LDR 30 C° Termistor 500 lux Fototransistor	Temporizador 555 Temporizador dual 555 Amplificador operacional
Switches	**Resistores**	**Inductores**	**Series 7400**
Botón push Interruptor	1K Resistor	1H Inductor Transformador 10:1	



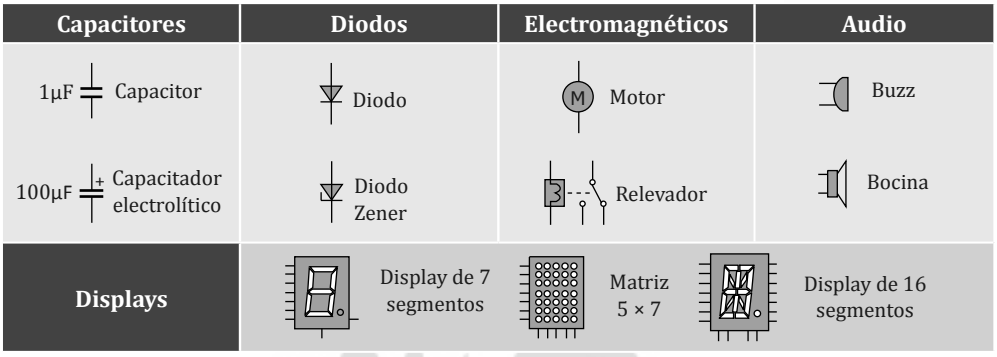


Figura 1.11 Simbología de los componentes electrónicos más representativos

B. Los dispositivos electrónicos

Estos elementos, por lo general, van desde microcontroladores, microprocesadores hasta placas de circuito impreso funcionales como *shields* (módulos o tarjetas de expansión), tarjetas adaptadoras, *mainboards*, placas de hardware libre (desarrollo) y herramientas electrónicas (medición, diagnóstico, análisis, muestreo y cálculo).

Muchos de los dispositivos electrónicos existentes en el mercado, a menudo, son utilizados para el diseño de proyectos. A continuación, se describen ciertos dispositivos electrónicos utilizados en el rubro de la electrónica digital y la ingeniería de hardware:

a. **Plataformas de desarrollo:** Se trata de placas diseñadas por algunos fabricantes de dispositivos electrónicos que le permiten al usuario trabajar con el ensamble de circuitos y programación de módulos. Este tipo de placas son ideales para comenzar a desarrollarse en el diseño, la programación de microcontroladores y la planeación de proyectos. Más adelante en el tema plataformas de hardware libre, de este mismo capítulo, se describen algunas de las plataformas de desarrollo más populares.

b. ***Shields*:** Son dispositivos que funcionan como extensiones de las distintas plataformas del mercado como es el caso de Arduino. Una *shield* tiene la finalidad de expandir el número de funciones que tiene el dispositivo origen. Estos elementos van desde placas auxiliares para el desarrollo de circuitos, placas de secuencia de led (Marleyno), dispositivos USB, tarjetas SD, módulos inalámbricos (XBee, bluetooth). Una *shield* es el equivalente en electrónica a las tarjetas de expansión para PC.

c. **Dispositivos de entrada y salida de datos:** Algunos ejemplos de dispositivos de entrada son las pantallas matriciales LCD, pantallas OLED y todo aquel medio que permite la visualización de información. Algunos dispositivos de salida empleados en el desarrollo de proyectos son, por lo general, teclados numéricos o alfanuméricos.

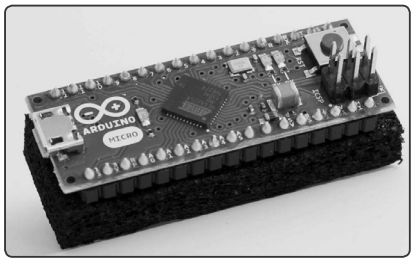


Figura 1.12 *Ejemplo de plataforma de hardware libre: Arduino Micro.*

Nota: Para realizar la presente actividad, debe ejecutar PCB Wizard y LiveWire. Posteriormente, abra los ejemplos que a continuación se solicitan, los cuales se hallan almacenados desde el momento de la instalación.

ACTIVIDAD 3

Dados los siguientes proyectos, liste y clasifique el material (componentes) que va a requerir para su construcción.

a) Sistema de monitoreo: _____

b) Contador de dos dígitos: _____

c) Amplificador de audio: _____

d) Sensor de luz: _____

1.1.3 Microprocesadores y microcontroladores

Los microprocesadores, microcontroladores y PLD son conocidos simplemente como sistemas microprogramables. Algunos autores, como Víctor Rossano[1] los han incluido en categorías como componentes electrónicos o dispositivos; aunque, a decir verdad, se trata de algo más que eso, pues estos elementos son capaces de superar el número limitado de funciones de un simple componente, o superar el grado de control que tiene un dispositivo (el cual fue desarrollado con el fin de satisfacer una necesidad particular). Los sistemas de microprogramación están encaminados a la innovación, la convergencia y el diseño de proyectos. Son capaces incluso de controlar un sistema entero de ordenadores o servidores.

El **microprocesador** es un elemento capaz de funcionar de manera integral como lo hace el cerebro humano. Algunos autores lo refieren como dispositivo electrónico, aunque, a decir verdad, se comportan más como un componente, ya que representan una unidad para el funcionamiento de cualquier dispositivo susceptible al control. Un microprocesador, por ejemplo, es el encargado de procesar la información suministrada por un dispositivo de entrada, y el encargado de proporcionar un pulso de salida mediante la realización de operaciones. Los microprocesadores trabajan a ciertos ciclos de frecuencia, y actualmente son utilizados en muchos dispositivos informáticos como tabletas electrónicas, computadoras, dispositivos móviles, dispositivos de red, etc.

Un **microcontrolador** es un circuito, el cual se encuentra mejor equipado que el microprocesador. Este se caracteriza por estar integrado por interfaces de conexión y tener la capacidad de controlar mayor número de componentes, e incluso dispositivos. Son conocidos como circuitos integrados programables (PIC). En el mercado electrónico existe una gran variedad de microcontroladores. En este libro se explica, más adelante, el modo de programar un microcontrolador ATMega328.

[1] Autor mexicano egresado de ESIME del IPN. Ha escrito de diversos libros sobre electrónica digital y microcontroladores PIC.

1.1.4 La importancia de las matemáticas y la física

Es muy común hacer uso de ciertas disciplinas o ciencias aplicadas como las matemáticas y la física para comprender el funcionamiento de la electrónica digital. Por un lado, el uso de las matemáticas resulta imprescindible tanto para la comprensión del funcionamiento de ciertas magnitudes o valores como la realización de algunos cálculos. La física, por su cuenta, establece la forma del comportamiento de fenómenos acorde a los materiales (el cual implica la manipulación de cargas) que lo componen y las leyes naturales que lo rigen.

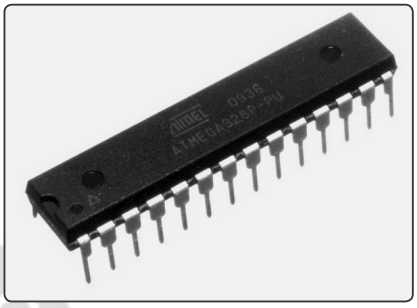


Figura 1.13 Microcontrolador de la compañía ATMEL: ATMega328

Otro campo que no se puede dejar a un lado es la informática, ya que gracias a ella se recurren a métodos como la programación y el desarrollo, lo que implica el control de ciertas interfaces con el uso de la computadora o herramientas computacionales.

A. Tipos de circuitos

Por lo general existen tres tipos de circuitos utilizados en la electrónica, se trata del circuito en serie (el cual puede ser implementado con una compuerta AND), el circuito en paralelo (el cual puede ser implementado con una compuerta OR), los circuitos mixtos (combinación de un circuito serie y paralelo).

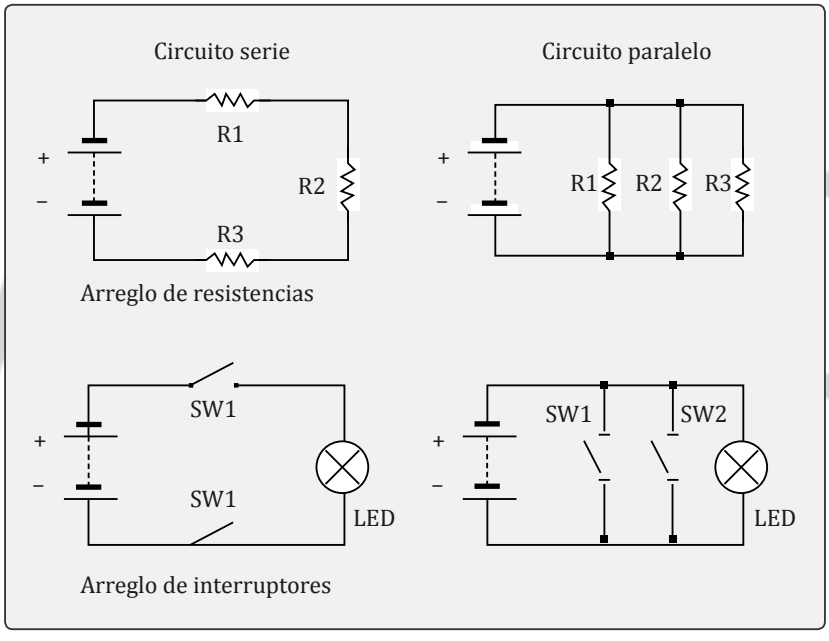


Figura 1.14 Tipos de circuitos: circuito en serie y circuito en paralelo

Como se puede apreciar, el diseño de estos circuitos, a menudo, requiere de algunos cálculos matemáticos, así como de ciertos material electrónicos, como una fuente de alimentación (3.3 V, 5 V, 9 V, 12 V, etc.), ledes, interruptores, resistencias, etc.

Los cálculos que se requieren están dados según algunas leyes de la física y la electricidad, las cuales se verán en el siguiente tema.

Nota de interés

Recurso interactivo de la ley de Ohm

Para hacer más interactivo el diseño de circuitos en serie, paralelo o mixto, se puede hacer uso de algunas herramientas de software o herramientas *online*. Una de las más populares es OHM ZONE, la cual se puede consultar desde su portal web: http://www.article19.com/shockwave/oz.htm.

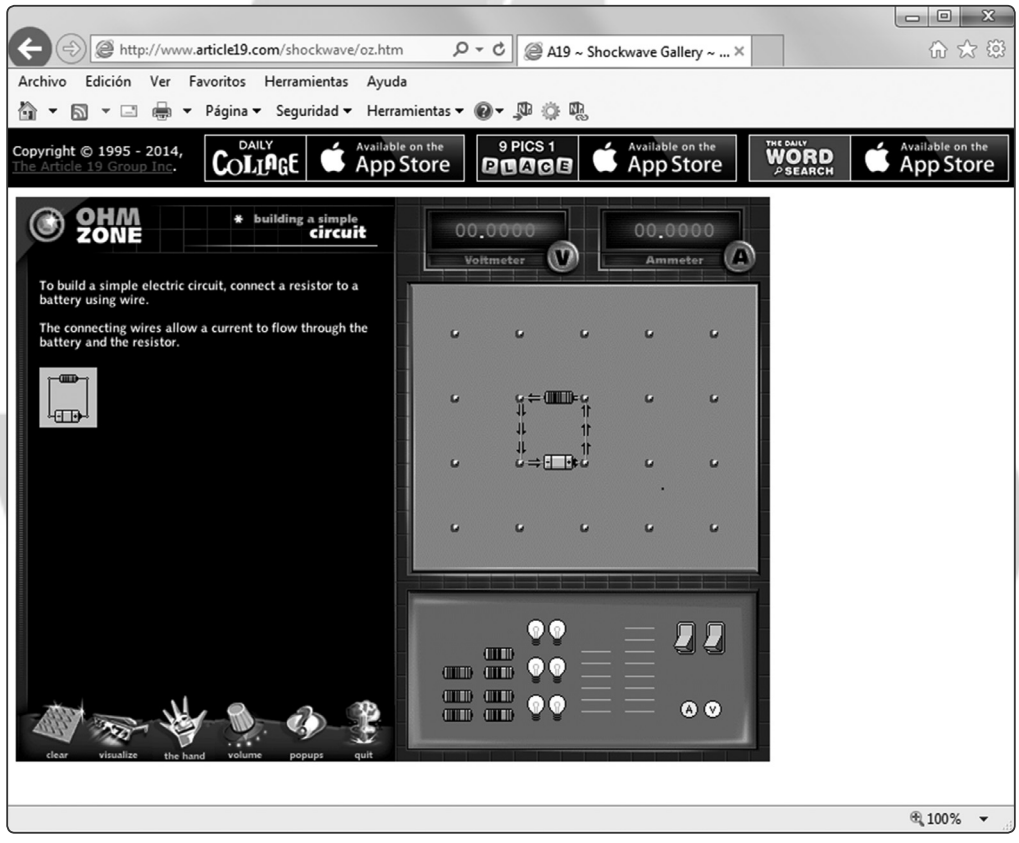


Figura 1.15 OHM ZONE, una herramienta interactiva *online* sobre manejo de circuitos en serie y paralelo

B. La ley de Ohm y la ley del Watt

Se trata de leyes de la física y la electricidad. La ley de Ohm, llamada así en honor a Georg Simon Ohm, establece el cálculo de la diferencia de potencial (V), la intensidad de corriente (I) y la resistencia en un circuito. Lo anterior se puede obtener en base al análisis del «triángulo de la ley de Ohm». La ley de Watt (potencia eléctrica), llamada así en honor a James Watt, por su parte, establece un cálculo de la potencia eléctrica en base al voltaje y la intensidad de corriente suministrada en un circuito.

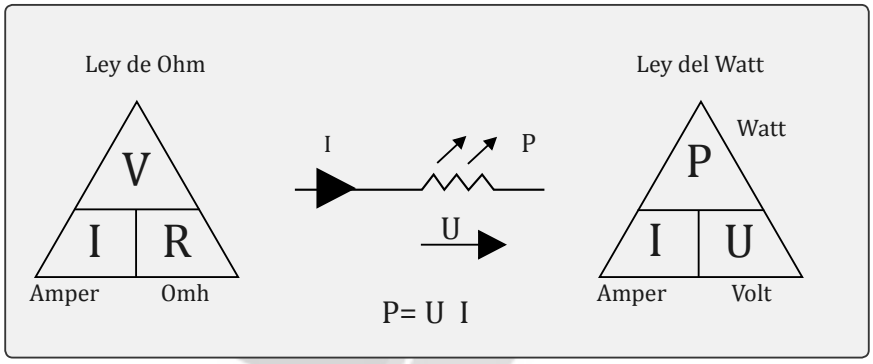


Figura 1.16 Triángulo de la ley de Ohm y la ley de Watt

Se puede apreciar que resulta muy sencillo obtener la operación adecuada para efectuar los cálculos antes mencionados. Las expresiones quedarían de la siguiente forma:

$V = I \times R$

$I = V / R$

$R = V / I$ (para la ley de Ohm)

$P = I U$

$U = P / I$

$I = P / U$ (para la ley de Watt)

$E = P \times t$ (energía es igual a potencia por tiempo)

Para el cálculo de voltaje, intensidad de corriente y resistencia total en un circuito, se debe considerar un parámetro adicional en la fórmula estándar utilizada, se trata del sufijo t. De tal modo que dicha fórmula quedaría de la siguiente forma: Vt = It Rt, según sea el caso.

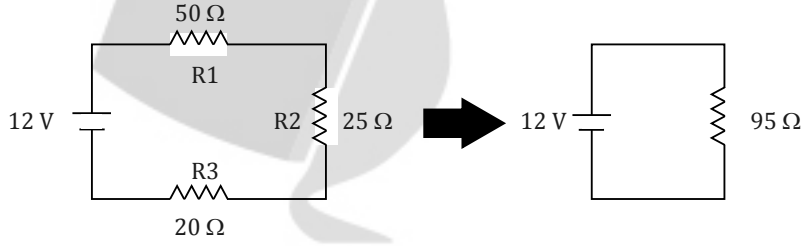
La unidad de medida del voltaje (V) es el **Volt**; para la intensidad de corriente (I), el **Amper**; y para la resistencia (R), el **Ohm**. Como se puede apreciar la potencia se mide en Watt.

Lo anterior es indispensable, sobre todo cuando el usuario trabaja con el análisis de circuitos electrónicos y eléctricos; con ello será capaz de determinar el número de resistencias que debe tener su circuito (así como el número de Ohmios por resistencia, la cual se determina por un código de colores), la intensidad de la corriente que fluye y, desde luego, el voltaje necesario para el funcionamiento de circuito.

Gracias a las nuevas tecnologías, actualmente ya es posible descargar apps que tienen como finalidad hacer más sencilla la tarea de cálculo de las variables anteriores; estas se pueden descargar de la página de Google Play.

Para comprender tanto la ley de Ohm como la ley de Watt, se propone a analizar el siguiente ejemplo:

a. Dado el siguiente circuito en serie, calcule la resistencia total (Rt), la intensidad de corriente total (It), la tensión de cada receptor (V1…Vn) para obtener la comprobación del voltaje total suministrado (Vt = 12V).



Para el cálculo de la resistencia total (Rt) se necesita sumar el valor de cada resistencia presente en el circuito, lo que implica el uso de la fórmula:

Rt = R1 + R2 + R3... Rn

Rt = R1 + R2 + R3 = 50 + 25 + 20, por lo tanto, la Rt = 95 Ohmios.

Para el cálculo de la intensidad de corriente total (It) que fluye en el circuito se emplea:

It = Vt/Rt = 12/95 = 0.126 A

Dada la intensidad total, se calcula la tensión (V) de cada receptor como se aprecia a continuación:

V1 = I1 × R1 = 0.126 × 50 = 6.3 V

V2 = I2 × R2 = 0.126 × 25 = 3.1 V

V3 = I3 × R3 = 0.126 × 20 = 2.5 V

Por último, solo falta comprobar si la suma de las tensiones es igual a la tensión total:

Vt = V1 + V2 + V3 = 6.3 + 3.1 + 2.5 = 11.9 V = 12 V

b. Dado el circuito anterior, calcule la potencia total del circuito y de cada receptor.
Para efectuar esta operación, debe considerarse la fórmula de la ley de Watt para el cálculo de la potencia total, como se muestra a continuación:

Pt = Vt × It = 12 × 0.126 = 1.5 W

P1 = V1 × I1 = 6.3 × 0.126 = 0.79 W

P2 = V2 × I2 = 3.1 × 0.126 = 0.39 W

P3 = V3 × I3 = 2.5 × 0.126 = 0.31 W

Para su comprobación, se efectúa la siguiente operación (suma de potencias de cada receptor):

Pt = P1 + P2 + P3 = 0.79 + 0.39 + 0.31

c. Calcule la energía total consumida por el circuito.
Para el cálculo de la energía consumida (Et) por el circuito, se hace uso de la fórmula: **Et = Pt x t**.

Et = Pt x t = (1.5 W) (3 horas) = 4.5W/h (vatios por hora). Ahora, si se desea obtener el cálculo de kilovatios por hora (Kw/h), se debe dividir la Pt entre mil.

Pt = 0.0015 × **3 = 0.0045Kwh**

Para cada receptor quedaría:

E1 = P1 × t = 0.79 × 3 = 2.37 Wh

E2 = P1 × t = 0.39 × 3 = 1.17 Wh

E3 = P1 × t = 0.31 × 3 = 0.93 Wh

Para su comprobación, se efectúa la siguiente operación:

Et = E1 + E2 + E3 = 2.37 + 1.17 + 0.93

Resultados obtenidos:

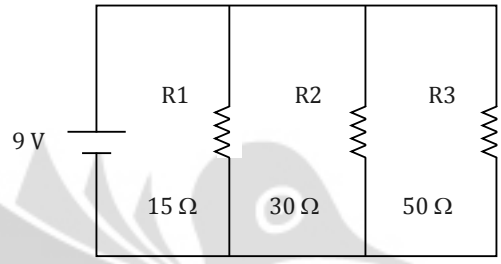
Rt= 95 Ohmios, It = 0.126 Amperes, Vt= 12 Voltios

Pt = 1.5 Watt

Et = 4.5 Watt por tres horas

Ejemplo de aplicación de la ley de Ohm y la ley de Watt en un circuito en paralelo

a. Dado el siguiente circuito en paralelo, determine el voltaje total (Vt), calcule la intensidad total (It) y de cada receptor (I1, I2, I3…In).



En un circuito en paralelo, la tensión total dada (en este caso 9V) será la misma para cada receptor, es decir: **Vt = V1 = V2 = V3…Vn** = 9 Voltios.

La intensidad de corriente total (It) se calcula aplicando la siguiente fórmula:

It = I1 + I2 + I3…In

Por lo tanto, se debe calcular de manera previa la intensidad de corriente para cada receptor como se ve a continuación:

I1= V1/R1 = 9/15 =0.6 Amperios

I2= V2/R2 = 9/30 =0.3 Amperios

I3= V3/R3 = 9/50 =0.18 Amperios

Por lo tanto, It = I1 + I2 + I3…In = 0.6 + 0.3 + 0.18 = 1.08 Amperios

b. Dado el circuito anterior, obtenga el valor de la resistencia total (Rt).

Para esta tarea se emplea la siguiente fórmula:

Rt = 1/ (1/R1) + (1/R2) + (1/R3) = 1/ (1/15) + (1/30) + (1/50) = 1 / (0.066) + (0.033) + (0.02) = 1/ 0.119 = 8.40 Ohmios

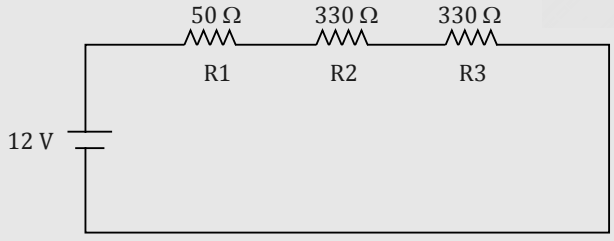
c. Compruebe el valor total de la tensión suministrada (en este caso 9 voltios).

Para comprobar el voltaje total (Vt) del circuito se efectúa la operación **Vt = It x Rt** establecida por la ley de Ohm.

Vt = It x Rt= 1.08 x 8.40 = 9.0 V

ACTIVIDAD 4

1. Dado el siguiente circuito en serie, obtenga:

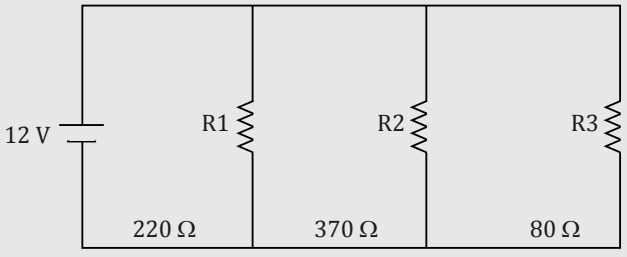


a) Resistencia total (Rt), intensidad de corriente total (It) y tensión de cada receptor (V1…Vn). Efectúe las comprobaciones necesarias.

b) Cálculo de la potencia total del circuito y de cada receptor.

c) Cálculo de la energía total consumida por el circuito.

2. Dado el siguiente circuito en paralelo, obtenga:



a) Voltaje total (Vt), intensidad total (It) y de cada receptor (I1, I2, I3…In).

b) Valor de la resistencia total (Rt).

c) Comprobar el valor total de la tensión suministrada.

d) Potencia total del circuito y de cada receptor.

e) Energía total consumida por el circuito para 4 horas.

C. Código de resistencias

Actualmente, es muy habitual el uso de resistencias al momento de trabajar con circuitos electrónicos. Estos componentes pasivos por lo regular están fabricados de carbón y tienen como objetivo oponerse al paso de la corriente. La resistencia es muy utilizada para la conexión tanto de entradas (*switches*) como de salidas (led).

Toda resistencia posee un conjunto de líneas de colores trazadas en forma vertical (llamadas bandas), las cuales sirven para identificar su medida en Ohmios. Las resistencias pueden ser de diversos tamaños (que hoy forman parte de un estándar) y no poseen polaridad.

Existen tablas con insignias de colores que permiten identificar la cantidad de Ohmios que poseen, estas se pueden descargar de la web, adquirir en tiendas de electrónica (a modo de calcomanía, calendario o boceto).

Nota de Interés

Calculadora de resistencias *online*

Actualmente existen también diferentes recursos *online* que permiten al usuario calcular el valor de una resistencia en Ohmios. Un ejemplo de este recurso se puede consultar a través del siguiente enlace:
http://www.digikey.com/es/resources/conversion-calculators/conversion-calculator-resistor-color-code-5-band.

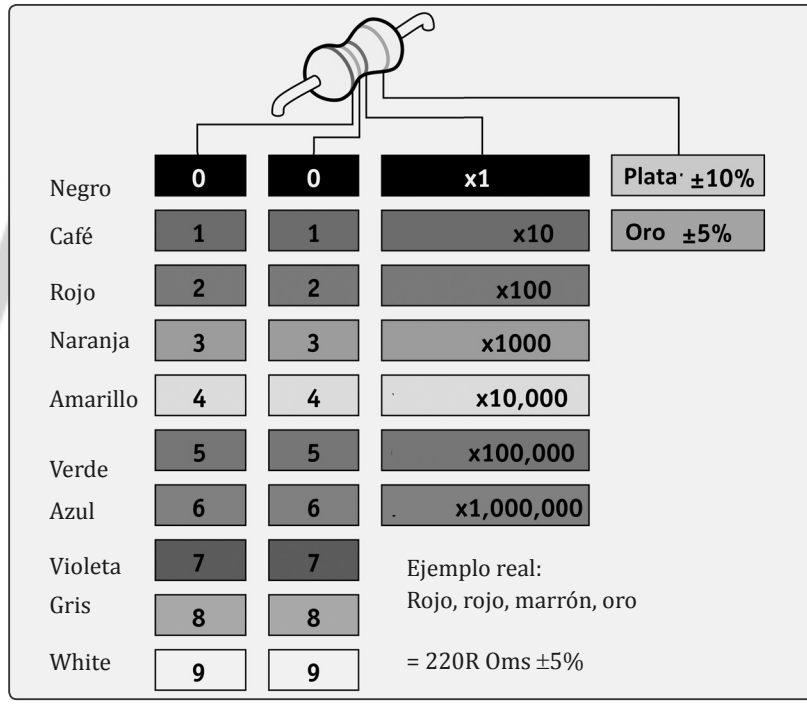


Figura 1.17 Código de colores de una resistencia

El cálculo de una resistencia idónea para cualquier circuito (que dependerá del arreglo de componentes) se puede efectuar utilizando la siguiente fórmula:

$$R = (VCC – Vf)/If$$

Donde:

R = Resistencia limitadora (expresada en Ohm)

VCC = Tensión de alimentación (voltaje de la batería)

Vf = Caída típica de voltaje del led (la cual oscila entre 1.2 V hasta 5 V)

If = Intensidad de corriente típica

Ejemplo de cálculo de resistencia

Se posee un circuito electrónico cuyo voltaje de alimentación es de 5V y se desea colocar un led con Vf = 1.2V, cuya intensidad de corriente o If = 20mA (miliAmperes). Su resistencia limitadora será de 190 Ohmios. Es necesario tomar en cuenta que dicho valor es aproximado y, por lo tanto, está permitida la colocación de resistencias con valores equivalentes, esto sin alterar el funcionamiento del circuito. En caso de rebasar un límite de 100mA, el led se fundirá.

1.2 PLATAFORMAS DE HARDWARE LIBRE

Actualmente existe una gran variedad de plataformas de desarrollo, las cuales han adoptado incluso la filosofía del hardware libre y el código abierto. La mayoría de estas plataformas permite el desarrollo de proyectos electrónicos, proyectos de domótica, e interesantes proyectos de robótica. Algunos ejemplos de estas plataformas son: la placa Arduino, la interfaz Galileo de Intel, *hardkernel* ODROID-U, *launchpad*, DCcduino, etc.

1.2.1 La plataforma Arduino

Una de las plataformas más populares es sin duda la placa Arduino, la cual ofrece un entorno de desarrollo muy amigable y fácil de comprender, ya que su programación se realiza en lenguaje C. La placa Arduino se presenta en diferentes diseños según las necesidades del usuario, estos son descritos en el portal de Arduino (www.arduino.cc) en la sección productos. Del sitio de la plataforma se puede descargar desde el software necesario para su configuración inicial hasta ejemplos para poner a prueba su potente versatilidad[2].

[2] Para mayor detalle sobre la plataforma Arduino, véase Germán, T. (2014).

En la siguiente infografía, se describe de manera breve cada una de las partes de una placa Arduino. Para tal efecto se utilizará el esquema de Arduino Genuino UNO.

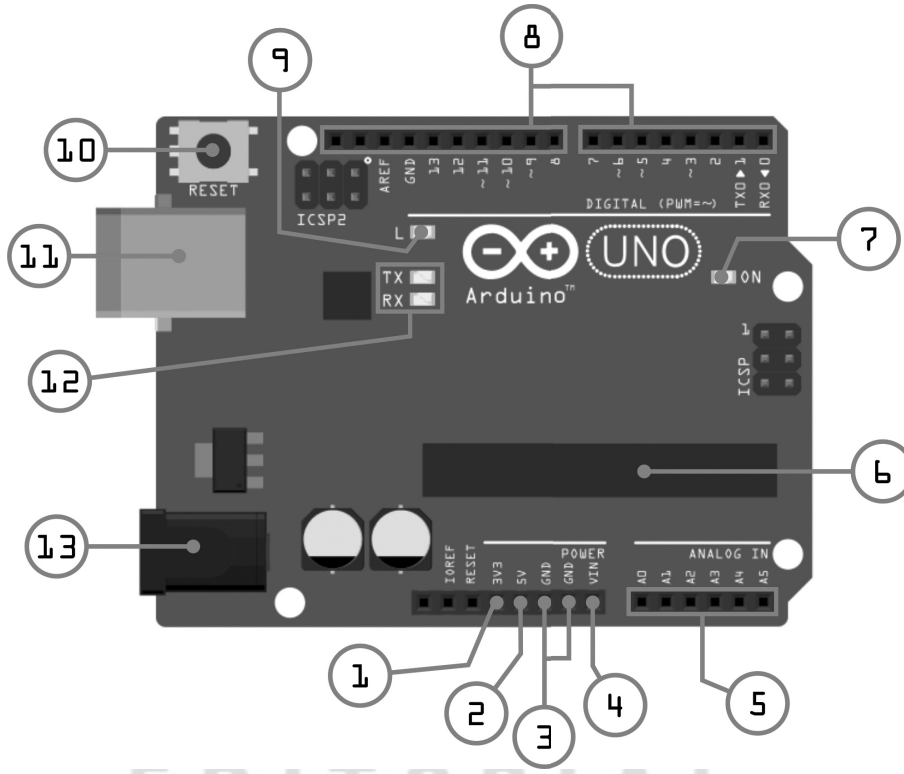


Figura 1.18 Infografía de las partes que conforman Arduino UNO

N.°	Componente	Descripción
1	3V3 o 3.3V	Se trata de un pin capaz de entregar una tensión de 3.3V. Ideal para alimentar componentes electrónicos que requieran este voltaje para funcionar
2	5V	Se trata de una tensión regulada por Arduino, la cual es entregada por la interfaz USB de la plataforma
3 y 4	GND	Este pin se utiliza para conectar los diferentes componentes electrónicos o dispositivos a tierra
5	Pines analógicos	Se trata de un conjunto de pines que sirven para representar una señal analógica como magnitud, capaz de tomar valores de un intervalo de 0V-5V (1024 niveles)
6	Chip ATMEL serie ATMega238	Se trata de un microcontrolador de la marca ATMEL que representa el núcleo de placa. Es el encargado de procesar la información
7	Led ON/OFF de Arduino	Consiste en un led indicador de flujo de voltaje para la placa Arduino. Le permite al usuario detectar si el dispositivo está conectado y funcionando

8	Pines digitales	Por lo regular se hallan rotulados del 0 al 13 y le permiten al usuario tanto escribir como recibir información digital. Estos pines deben ser configurados y programados para su funcionamiento. Los pines digitales identificados con el símbolo «~» son pines capaces de generar señales con modulación de ancho de pulso (PWM)
9	Led Pin 13	Se trata de un led de prueba, a menudo usado para verificar el funcionamiento de pines digitales. Es de color naranja y se conecta al pin 13. Incluye una resistencia ideal para efectuar pruebas desde *protoboard*
10	Botón reset	Permite reiniciar la placa. Muy útil cuando el dispositivo ha quedado congelado ya sea por software o por hardware
11	Conector USB tipo B	A través de este conector en serie se suministra un voltaje de 5V a la placa Arduino. Permite también la comunicación entre la PC y la plataforma
12	Led RX y TX	Son ledes que indican cuándo se efectúa tanto la recepción (RX) como la transmisión de datos (TX)
13	Conector de voltaje AC	Se trata de un conector para la alimentación de una placa Arduino. El voltaje que soporta es de 5V hasta 20V

Para conocer más sobre el funcionamiento de la placa Arduino UNO mediante algunos ejemplos, se propone consultar el capítulo siete de este libro.

1.2.2 La plataforma Raspberry Pi

Raspberry Pi es considerada, además de una plataforma de hardware libre, como una MiniPC. Su formato de forma es pequeño, pero muy similar a una computadora convencional. Esta se conforma de puertos e interfaces de entrada y salida llamado *Onboard*. En este se puede conectar una pantalla, un teclado, un mouse y una serie de dispositivos de entrada y salida de datos. Tiene un sistema operativo propio llamado Raspbian, originalmente basado en la distribución Debian de GNU-Linux[3]. Gracias a este tipo de arquitectura se pueden desarrollar proyectos informáticos, proyectos electrónicos, proyectos de domótica, etc.; incluso se pueden combinar con otras plataformas como Arduino[4].

[3] GNU Linux consiste en un sistema operativo libre creado por Linus Torvalds y Richard Stallman. Representa una alternativa a Microsoft Windows y a menudo es conocido como el sistema del pingüino.

[4] Para conocer algunos proyectos de desarrollo con el uso de la arquitectura Raspberry Pi en combinación con Arduino, revísese el libro «Electrónica: plataformas Arduino y Raspberry Pi».

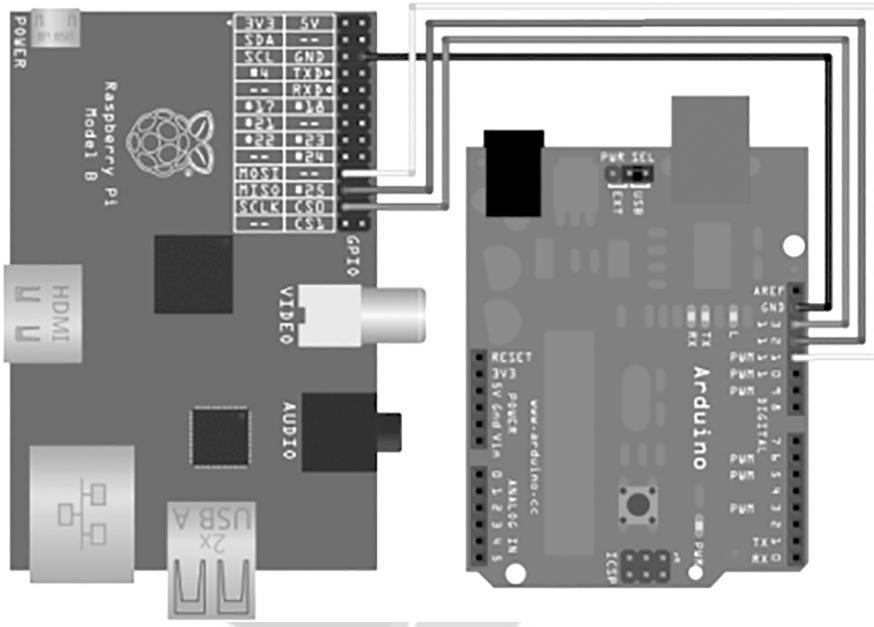


Figura 1.19 Esquema de conexión entre Arduino y Raspberry Pi

ACTIVIDAD 5

1. Resuelva los siguientes problemas sobre cálculo de resistencia. Para su comprobación será necesario verificar la tabla de resistencias estándar. Para ello, utilice el recurso *online*, que se hace referencia en el tema anterior, código de resistencias.

   a) Se posee un circuito electrónico, cuyo voltaje de alimentación es de 9 V, y se desea colocar un led con Vf = 1.2 V, cuya intensidad de corriente o If = 20 mA. ¿Cuál es el valor adecuado de su resistencia limitadora?, ¿cuál es el código de colores de dicha resistencia?

   Datos Fórmula Sustitución Resultado

   Código de colores: _____

b) Se requiere montar un circuito electrónico cuyo voltaje de alimentación es de 5 V y se desea colocar un led con Vf = 1.5 V, cuya intensidad de corriente o If = 20 mA. ¿Cuál es el valor adecuado de su resistencia limitadora?, ¿cuál es el código de colores de dicha resistencia?

Datos Fórmula Sustitución Resultado

Código de colores: _____

2. ¿Cuál es el valor apropiado de las resistencias que requieren los siguientes montajes realizados en Arduino?

a) Secuencia de 4 ledes de 5 mm:

_____

b) Encendido de un led, mediante el uso de un interruptor:

_____

c) Un sumador binario de 4 bits (CI-7483A):

_____

3. ¿Cuál es el número de bandas y código de colores correspondientes de los siguientes valores de una resistencia?

a) Resistencia de 330 Ohmios → Número de bandas: _____, colores:

_____

b) Resistencia de 470 Ohmios → Número de bandas: _____, colores:

_____

c) Resistencia de 1 K Ohmios → Número de bandas: _____, colores:

_____

d) Resistencia de 10 K Ohmios → Número de bandas: _____, colores:

_____

1.3 PROGRAMACIÓN DE HARDWARE

Por lo general para programar las funciones de un sistema microprogramable[5] (en caso del microcontrolador o PIC) es necesario contar con un lenguaje de programación apropiado, además de un editor o entorno de desarrollo, habitualmente proporcionado por el fabricante (como es el caso de los microcontroladores PICAXE y su famoso PICAXE Programming Editor).

Algunos ejemplos de lenguajes de programación para microcontroladores son: BASIC, MikroC, lenguaje ensamblador, Simulink y lenguaje C, siendo este último uno de los más empleados en la actualidad, del cual se habla más adelante.

Actualmente, muchas de las funciones de un microcontrolador (para la manipulación de módulos de hardware) pueden crearse mediante el uso de cualquier interfaz gráfica de programación (utilerías de software que emplean por lo regular diagramas de flujo). Una de las herramientas más populares en este rubro es FlowCode (para su consulta se precisa revisar su sitio de Internet: https://www.matrixtsl.com/flowcode). Algunos ejemplos de otros entornos de desarrollo o editores (que en ocasiones dependen directamente del fabricante, serie o tipo de PIC) son: Niple, ColdFire, MPLAB, eZ System, Matlab (que utiliza el lenguaje de programación Simulink).

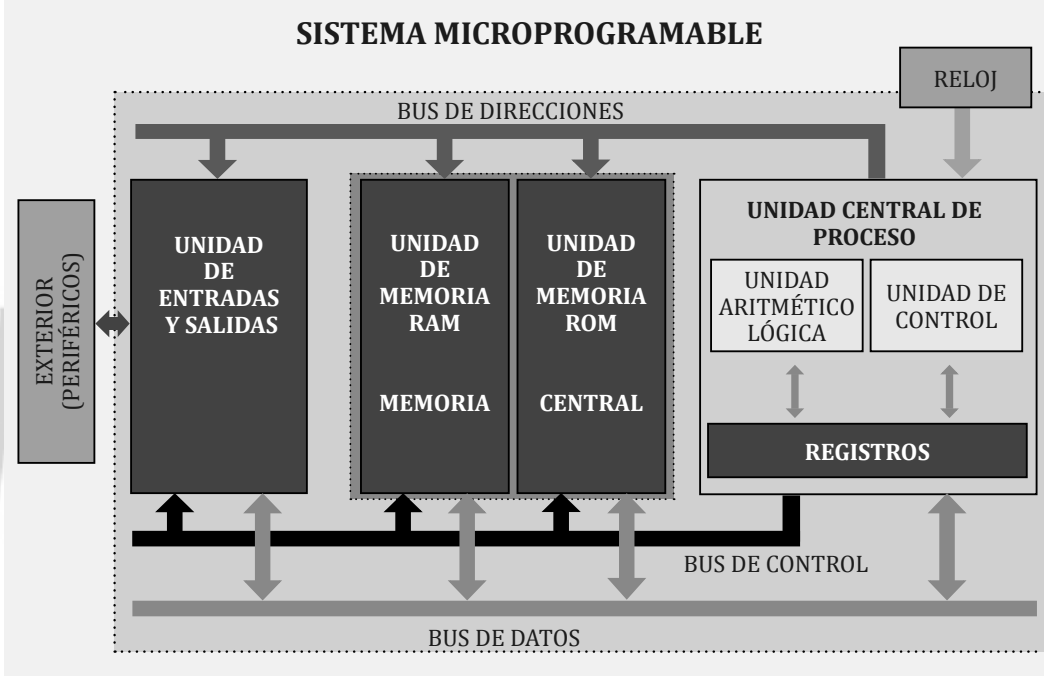


Figura 1.20 Esquema que muestra las partes internas de un sistema microprogramable

[5] Los sistemas microprogramables son computadoras que integran elementos tales como interfaces de entrada, memoria, unidades centrales de proceso e interfaces de salida.

El terreno de la ingeniería de hardware es inmenso, por ello mismo se aconseja la búsqueda y revisión de los esquemas de circuitería interna de los distintos PIC, módulos de programación o componentes electrónicos. Para tal efecto es necesaria la consulta de su respectivo *datasheet*[6], la cual puede ser consultada en Internet o en algunas ocasiones es proporcionada por el fabricante de la unidad.

Muchos fabricantes de módulos de hardware o PIC proporcionan, mediante su portal web, las herramientas necesarias para comenzar a desarrollar aplicaciones o proyectos novedosos, promocionando así el uso de sus productos. Incluyen, además, manuales, ejemplos, código fuente, controladores, compiladores o IDE de desarrollo, y, desde luego, las respectivas hojas de datos de cada producto. Algunos fabricantes son: Microchip Technology Inc (el cual pone en manos del usuario una herramientas de codificación y configuración de PIC llamada MPLAB CODE-CONFIGURATION, la cual se puede consultar a través de su sitio web), Dallas Semiconductors, AMTEL, Hitachi y Texas Instruments.

Es importante no confundir el lenguaje de programación con entornos de desarrollo, ni tampoco confundir lo anterior con herramientas de diseño de circuitos electrónicos y simulación (como es el caso de Proteus, LiveWire-PCB Wizard, LabView, Multisim, Electronic Workbench).

En resumen, para la programación de un módulo o, en su defecto, un circuito integrado programable (PIC) es necesario contar con una computadora personal, un entorno de desarrollo o IDE, una placa de prueba (cuya interfaz de conexión puede ser USB, mini-USB, micro-USB, etc.) la cual funciona como nexo entre la PC y el microcontrolador (a través de un medio de transmisión adecuado y por lo general en modo serie) y un paquete de controladores.

1.3.1 La importancia de lenguaje C en la ingeniería de hardware

En un principio, se utilizó lenguaje máquina para la comunicación entre dispositivos de hardware, con el paso del tiempo se han dado lugar a lenguajes de mediano nivel (lenguaje ensamblador) y actualmente a lenguajes de alto nivel como lenguaje C, del cual existen distintas variantes o evoluciones.

C es un lenguaje muy versátil, robusto, demandado y, por encima de muchos, el más intuitivo al momento de trabajar. Representa la base de lenguajes modernos como JAVA y C#. A lo largo de su existencia (desde los años 70, creado por Brian Kernighan y Dennis Ritchie en los laboratorios Bell de AT & T) ha sido muy utilizado no solo para la programación de hardware, sino también para el diseño de sistemas operativos para PC, móviles y otro tipo de dispositivos. Actualmente, se desarrollan infinidad de librerías o macros para este lenguaje, lo que ha hecho posible la integración de funciones adicionales al entorno.

[6] La *datasheet* es también conocida como hoja de datos. Generalmente incluye diagramas de conexión y la descripción de las partes que integran un circuito, componente o dispositivo.

1.3.2 Aplicaciones de lenguaje C en la electrónica digital

En la actualidad, la programación de placas Arduino incluyen el conocido lenguaje C para su programación. Por ello, suele ser uno de los favoritos de muchos usuarios. A continuación, se muestran dos ejemplos de código fuente desarrollado en C. A la izquierda se ilustra el funcionamiento de un voltímetro y a la derecha el control de un servomotor, ambos desde la plataforma Arduino UNO. Las presentes sentencias y funciones serán analizadas a detalle en el capítulo siete de este libro.

Ejemplo de aplicación de lenguaje C en la electrónica

```
//Creación de un Voltímetro
float voltaje;
void setup()
{
Serial.begin(9600);
}
void loop()
{
voltaje = analogRead(A4) / 1023.0 * 5;
Serial.println(voltaje);
delay(700);
}
```

```
//Control de un servomotor
#include<Servo.h>
#define pinServo 7 //todos los digitales
para servo
Servo servoMotor;

void setup()
{
pinMode(pinServo,OUTPUT);
servoMotor.attach(pinServo);
//unir servomotor     con el pin
servoMotor.wvrite(0);
//write= recibe
}
void loop()
{
ti=0,grados=0;i<3;i++,grados+=90)
{
servoMotor.write(grados);
delay(500);
  }
}
```

En la codificación del programa de la derecha, se puede apreciar la existencia de la siguiente línea de código: **#include<Servo.h>**. Se trata básicamente de una librería, la cual debe ser declarada en cualquier código desarrollado en C. Una librería es un conjunto de instrucciones almacenadas en distintos archivos de programa que tienen como objetivo ejecutar una acción. En este ejemplo, la librería <Servo.h> manda llamar esa serie de archivos, los cuales fueron diseñados con el propósito de manipular un servomotor. La extensión de una librería declarada en C tiene la extensión h.

Se sabe que el hardware debe trabajar en conjunto con el software, pues ambos dependen uno del otro para arrojar un resultado esperado.

Para entender muchas de las instrucciones de lenguaje C, pero, sobre todo, para ser capaces de programar cualquier cosa, es necesaria tanto la lógica como el uso de las matemáticas. Para ello, resulta importante hacer hincapié en el estudio sobre manejo de operadores, cálculos y matemáticas en general (cálculo integral y diferencial, probabilidad y estadística, analítica, álgebra lineal-booleana, etc.). Pues gracias a eso, se pueden conseguir proyectos que incluyan sensores (temperatura, humedad, distancia, infrarrojo, movimiento, etc.), lo que implica el manejo de operaciones matemáticas en general.

A. Programación de dispositivos móviles

En la actualidad el campo de la programación de software tiene varias vertientes y, por lo tanto, diferentes campos de acción. Le toca el turno a los dispositivos como celulares y tabletas.

Todo dispositivo informático, como los anteriormente mencionados, son capaces de trabajar gracias a un sistema operativo (SO), el cual funge como plataforma para hacer posible la interacción con el usuario final. Algunos SO móviles actuales tienen una peculiaridad muy interesante, son de código abierto (el más famoso es Android, originalmente basado en GNU-linux), lo que permite el desarrollo de aplicaciones o extensiones (app). En caso de otros sistemas, aunque no son de código abierto, permiten ejecutar aplicaciones desarrolladas por terceros sobre su propio entorno. Para la creación de app para Android se recomienda el uso de App Inventor, Android Studio, App Makr y PhoneGap.

Actualmente, existen millones de aplicaciones que pueden ser descargadas de internet, algunas con cierto costo, y otras totalmente gratuitas. Pero, ¿qué tiene que ver la programación de dispositivos móviles con la electrónica digital o la ingeniería de hardware?, la respuesta es sencilla; para ello, es importante recordar que, para poder manipular, por ejemplo un brazo mecánico, o conseguir controlar el alumbrado de una casa habitación, o simplemente para hacer avanzar un robot diferencial de manera automática, se necesita disponer de una app en el dispositivo móvil, mismo que funge como monomando para el control de cualquier prototipo o proyecto funcional. Por lo tanto, solo resta conectar ambos elementos, para tal efecto debe emplearse una *shield* que proporcione una interconexión inalámbrica (Arduino Wifi Shield, XBee, bluetooth, WiFly shield, etc).

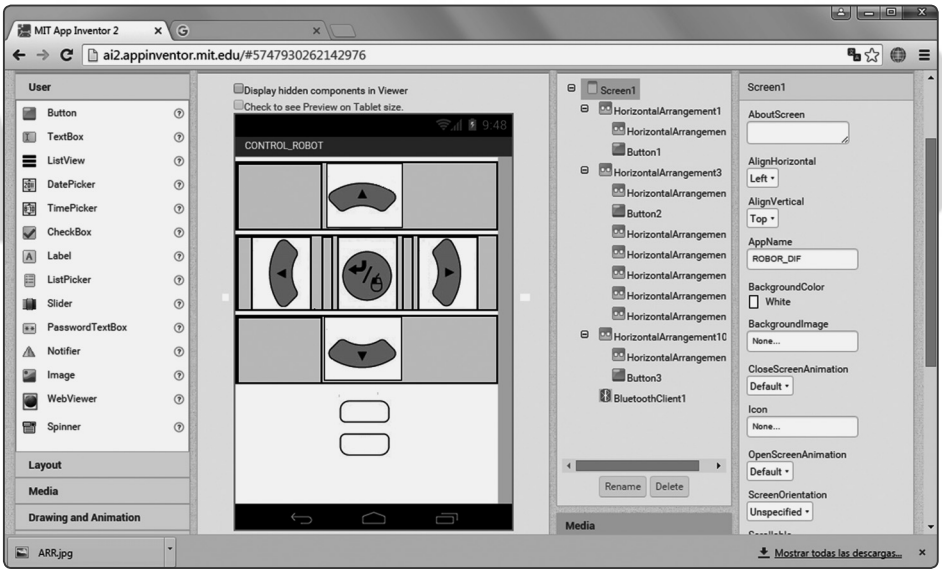


Figura 1.21 Ejemplo de app para Android creada desde App Inventor

Un ejemplo de robot diferencial son los carritos de control remoto. Estos incluso pueden ser controlados desde la PC que sirve para programarlos o desde los dispositivos móviles. Este tipo de robot a menudo utiliza sensores de distintos tipos, los cuales deben ser programados para conseguir el movimiento deseado en función de sus ángulos y ciertas matrices de dirección, de allí el concepto de robot diferencial.

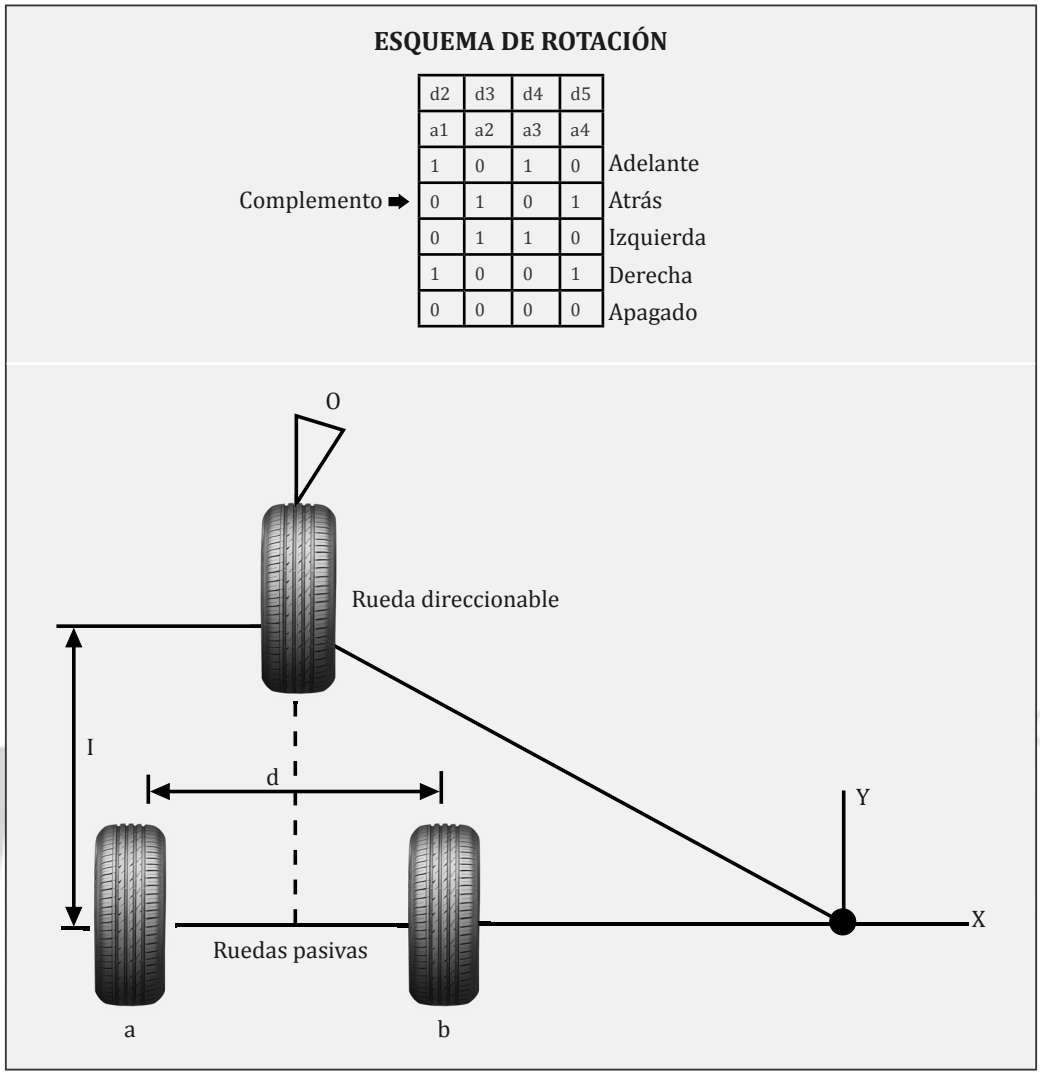


Figura 1.22 Esquema que muestra el funcionamiento de un robot diferencial

ACTIVIDAD 6

1. Realice un cuadro comparativo de los siguientes entornos de desarrollo para programar módulos de hardware.

Flowcode	Nipple	MPLAB

2. Realice un cuadro comparativo de los siguientes lenguajes de programación para PIC.

Lenguaje ensamblador	Simulink	Lenguaje C

3. Realice un cuadro comparativo de los siguientes simuladores de circuitos electrónicos.

LiveWire	Electronic Workbench	Proteus

4. Realice un cuadro comparativo de las siguientes herramientas de software para la creación de app con Android.

AppInventor	Android Studio	PhoneGap

1.4 MANEJO DE PUERTOS E INTERFACES DE LA PC

Los puertos e interfaces[7] de una computadora le permiten al usuario mantener una conexión física con cualquier otro dispositivo electrónico, placa o plataforma. Con anterioridad, el puerto paralelo (LPT1) era el más utilizado para la transferencia de datos. Representó un pilar para la comunicación e interacción con el hardware. Sin embargo, con el paso del tiempo este cayó en desuso, dando lugar al puerto USB (Universal Serial Bus), el cual es considerado como uno de los puertos más utilizados en la industria del hardware y la informática.

Se sabe que para la programación o configuración de un módulo de hardware, es necesario contar con un medio de transferencia de datos compatible con un puerto de conexión (generalmente un cable). El cual acepta un tipo de formato físico estándar de fabricación o tipo de interfaz. Actualmente, existen diferentes tipos de conexión o interfaz USB, como se ha mencionado (micro-USB, mini-USB), USB tipo A y USB tipo B. Muchas de las plataformas de hardware libre, hoy, hacen uso de este puerto para conseguir la comunicación con una PC.

Los medios de transmisión USB, además, sirven como nexo para alimentar el dispositivo en cuestión, pues es importante recordar que el voltaje máximo que proporciona un puerto USB es de 5V (nivel de tensión HIGH).

1.4.1 Interfaces de comunicación serial

La conexión entre distintos dispositivos puede hacerse mediante la adopción de alguna tecnología en particular. Actualmente se diseñan *shields*, placas o plataformas que hacen posible el envío (TX) y recepción de señales (RX). Recientemente se han desarrollado plataformas completas que fungen como módulos independientes y que se pueden conectar a un dispositivo o proyecto electrónico, ese es el caso de Arduino BT (Bluetooth).

A. Conexión inalámbrica

Las tecnologías inalámbricas aplicadas a la electrónica con mayor auge en la actualidad son: Wifi, bluetooth, XBee y las relacionadas con el rubro GSM. A continuación, se describen de manera breve:

a. Bluetooth: Por un lado se tiene la *shield* bluetooth, se trata de un pequeño módulo de hardware que tiene la finalidad de crear un enlace inalámbrico entre distintos dispositivos electrónicos. En otras palabras, permite la creación de redes inalámbricas de corta distancia. La distancia máxima permitida es de 3 a 5 metros. En el mercado existen distintos modelos de *shields* bluetooth, originalmente creados para Arduino, quienes han impulsado al mercado un módulo bluetooth similar al Arduino UNO, pero esta vez totalmente inalámbrico. Las características especificadas de este último lanzamiento son las siguientes:

[7] Para la obtención de información adicional acerca del tema de interfaces de una PC, véase Gilberto, G (2012).

El Arduino BT está basado en el ATmega168, pero ahora se suministra con el 328 y el módulo Bluegiga WT11 bluetooth. Es compatible con la comunicación serie inalámbrica a través de bluetooth (pero no es compatible con auriculares Bluetooth u otros dispositivos de audio).

Cuenta con 14 pines de entrada y salida digital (de los cuales 6 se pueden utilizar como salidas PWM, y se puede utilizar para restablecer el módulo WT11), 6 entradas analógicas, un oscilador de cristal de 16 MHz, terminales de tornillo para poder, una cabecera ICSP y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador y se puede programar de forma inalámbrica mediante la conexión Bluetooth.

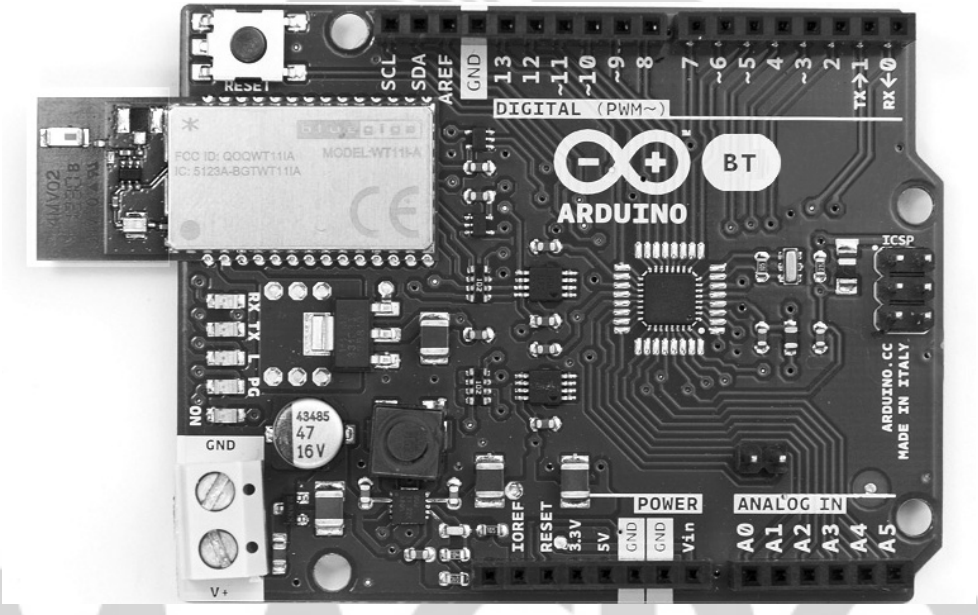


Figura 1.23 Arduino BT, plataforma que adopta la filosofía de comunicación serial de forma inalámbrica

b. Módulo XBee: Se trata de un chip electrónico de color azul, el cual se encuentra integrado por varios contactos de conexión. Este módulo es comúnmente usado para el intercambio de canales de emisión y recepción de datos entre dispositivos o proyectos. Los robots diferenciales son un ejemplo de aplicación de un módulo XBee. Existen diferentes series y modelos en el mercado, al igual que antenas para la comunicación.

Para obtener mayor información (tutoriales de configuración, descarga de software y datos técnicos) sobre este tipo de elementos, se recomienda la consulta del siguiente enlace: http://xbee.cl/que-es-xbee.

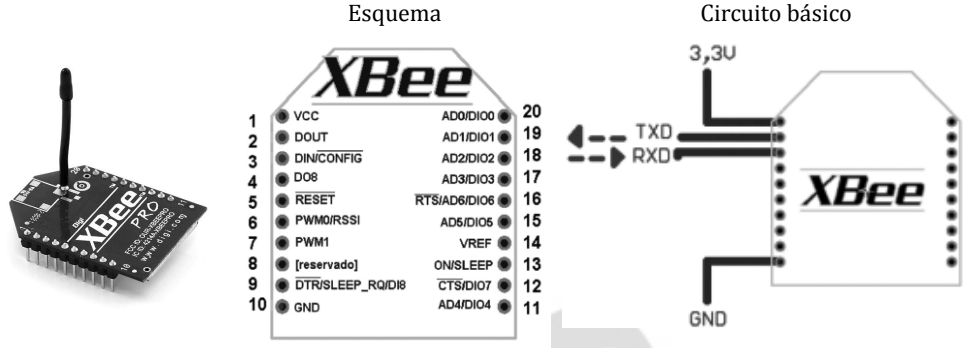


Figura 1.24 XBee, un chip que permite una comunicación serial inalámbrica entre dispositivos o proyectos

Para la configuración de uno o más módulos XBee, se recomienda el manejo de utilerías de software como X-CTU y la programación de canales RX y TX.

En la parte de abajo se muestra el código fuente de un par de programas desarrollado en lenguaje C. Ambos corresponden a la codificación del canal de transmisión y recepción respectivamente de un módulo XBee.

Ejemplo de codificación de un módulo XBee desde lenguaje C

```
//Código para transmisor (TX)

intpush = 2;

void setup()
{
Serial.begin(9600);
pinMode(push, INPUT);
}

void loop()
{
if (digitalRead(push) == LOW)
    {
Serial.print('A');
delay(200);
 }
}
```

```
//Código para receptor (RX)

int buzz = 5;
intledPin = 13;

void setup()
{
Serial.begin(9600);
pinMode(buzz, OUTPUT);
pinMode(ledPin, OUTPUT);
}

void loop()
{
if (Serial.available()>0)
{
if (Serial.read() == 'A')
{
digitalWrite(buzz, HIGH);
delay(20);
digitalWrite(buzz, LOW);
digitalWrite(ledpin, HIGH);
delay(30);
digitalWrite(ledpin, LOW);
   }
}
}
```

c. Módulo GSM: La finalidad de un módulo GSM es conectar cualquier plataforma de hardware a internet, utilizando la red inalámbrica GPRS (general packet radio service) o servicio general de paquetes vía radio. Para ello, se requiere, desde luego, un chip o tarjeta SIM. Su configuración para realizar y recibir llamadas o mensajes de texto es relativamente sencilla, aunque para ello se precisa el uso de un circuito de altavoz y un micrófono.

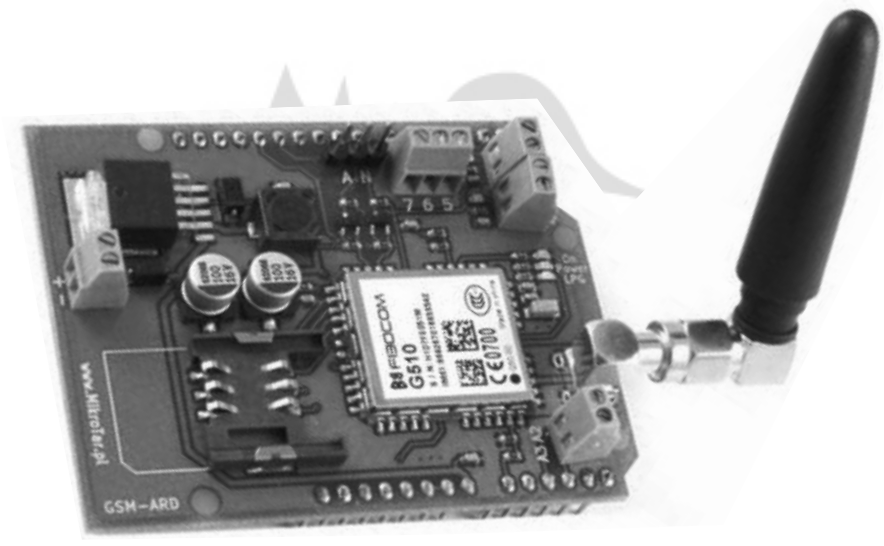


Figura 1.25 Los módulos GSM a menudo utilizan la red inalámbrica GPRS

1.5 SISTEMAS DE CONTROL Y ROBÓTICA

Como se ha mencionado, la electrónica tiene varios campos de acción, y sin duda el campo de los sistemas de control y la robótica no puede ser la excepción.

Para entender este contexto, es necesario comenzar a definir el término robot y sistema de control respectivamente.

Un robot consiste en un mecanismo capaz de aprender y por lo tanto ejecutar una serie de acciones previamente programadas, y que por lo general puede iniciar o terminar un proceso sin la intervención del usuario. Estos autómatas incorporan varios sistemas electrónicos que están encaminados a satisfacer una necesidad primaria, por ejemplo, el caso del robot o máquina industrial.

Un sistema de control es definido como mecanismo que posee la capacidad de controlar en tiempo real y en ambiente industrial, procesos secuenciales. A menudo los sistemas de control auxilian al usuario a simplificar procesos o tareas; encargándose incluso de coordinar un conjunto de sistemas autómatas. Dicho con otras palabras, los sistemas de control gestionan la puesta en escena de uno o más robots en un entorno.

Los robots no tienen que ser precisamente con apariencia humana, como se acostumbra a asociarlos, estos pueden presentarse al público en distintas formas: brazos, cajas, carros, etc. Sin embargo en lo que sí coinciden es que cualquiera que sea la forma que adopte un robot, puede hacer uso de partes mecánicas, componentes hidráulicos, electrónicos o alguna mezcla de estos.

Figura 1.26 Los robots industriales, programados y ensamblados para satisfacer necesidades específicas

1.5.1 Aplicaciones de la robótica

Actualmente, la robótica se encuentra encaminada al diseño de juguetes infantiles (carros, tráileres, helicópteros de control remoto, robots diferenciales, y todo aquel artefacto didáctico que incluya el manejo de sensores) o, incluso proyectos que tienen un propósito específico como es el caso de los drones, que son cada vez más frecuentes en el ámbito civil y que, originalmente, fueron desarrollados con fines militares. Otra aplicación vigente de la robótica se encuentra en la construcción de máquinas industriales (inyectoras automáticas, ensambladoras, troqueladoras), máquinas despachadoras (golosinas, refrescos, galletas, bebidas calientes o frías).

Figura 1.27 Los drones, ejemplo claro del uso de la robótica y la electrónica

1.5.2 Electrónica y domótica

Como se ha mencionado, un sistema de control se encarga de gestionar una serie de procesos que a menudo precisan el uso de componentes mecánicos o automatizados para alcanzar un objetivo. Estos sistemas se pueden apreciar incluso en el ámbito de la automatización o la domótica, término que se define como un conjunto de técnicas orientadas a la automatización de viviendas, oficinas, edificios y cualquier ambiente susceptible al control. La automatización de estos entornos implica la integración de sistemas de seguridad, tecnologías de gestión energética, comunicaciones, comodidad (simplificación de tareas), etc.

La **domótica**[8] emplea dispositivos electrónicos capaces de interconectar un conjunto de sistemas para obtener un resultado esperado. Hace uso de sensores capacitivos, inductivos, ópticos, de proximidad, entre otros.

Desde luego, que la electrónica y la ingeniería de hardware juegan un papel importante en la aplicación de técnicas de automatización, puesto que estos sistemas son programados para efectuar distintas tareas. El uso de plataformas como Arduino, incluso, permite al usuario el control de luces de una casa habitación, detectar el porcentaje de humedad presente en un jardín, detectar la temperatura ambiente de un aula de clases, determinar la distancia existente entre un automóvil y otro por la carretera, entre otros.

[8] Para obtener información sobre el tema de domótica, véase Cristóbal, R, Francisco & V, Carlos, C (2011).

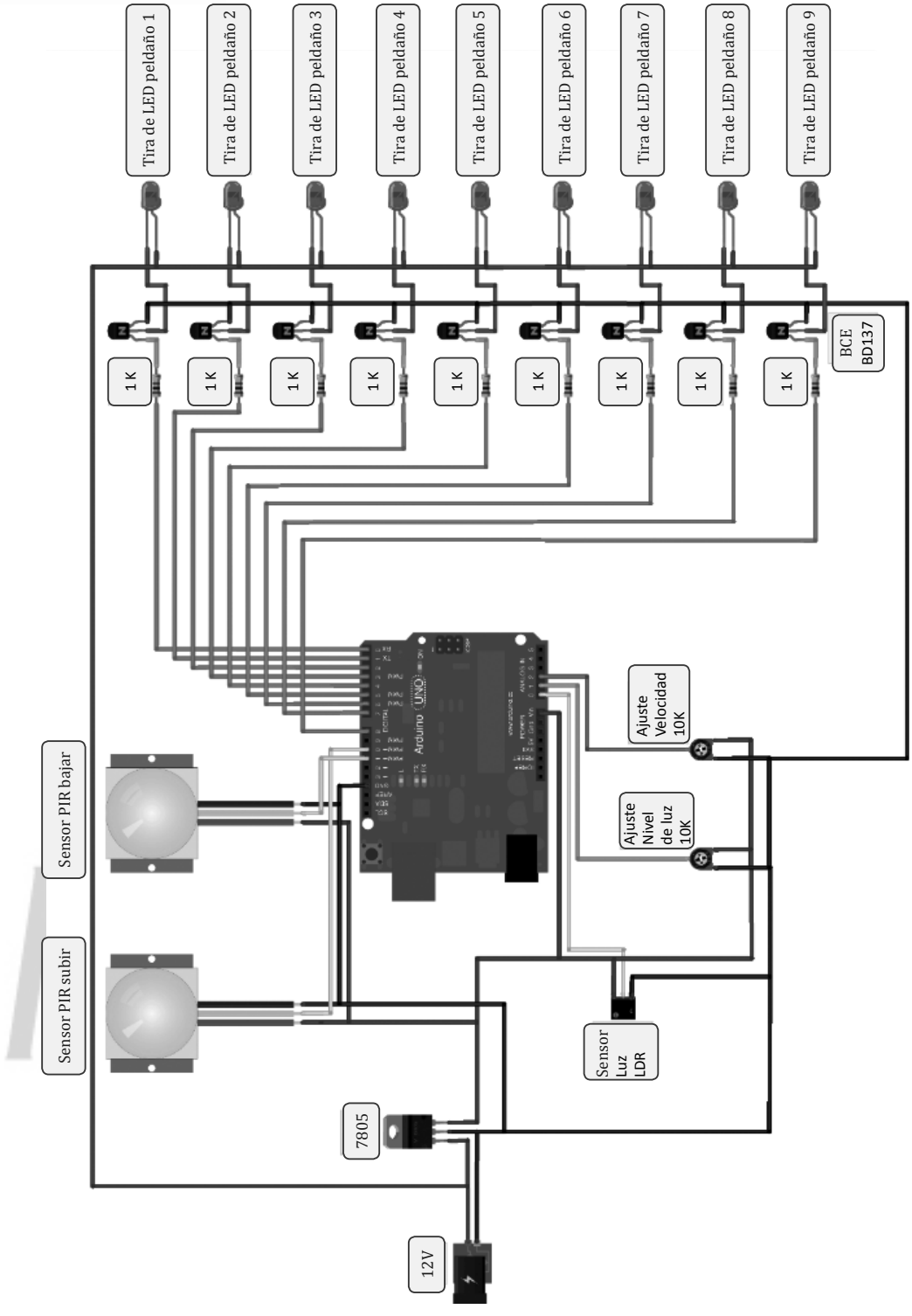


Figura 1.28 Esquema del control de luces ubicadas en la escalera de una vivienda (proyecto realizado con Arduino UNO)

ACTIVIDAD 7

1. ¿Cuáles son las características principales del lenguaje C?
2. ¿Cuáles son las partes que integran la estructura de un programa diseñado en lenguaje C?
3. ¿Qué es y qué características tiene un dron?
4. Describa el principio de funcionamiento de una *shield* bluetooth.
5. Investigue y describa de manera breve por lo menos tres proyectos desarrollados en Arduino.
6. Defina el término «domótica».

1.6 PLACAS DE CIRCUITO IMPRESO

Las placas de circuito impreso o PCB (*print circuit board*) son generalmente el recurso integrador de un sistema electrónico funcional. A menudo, la producción de estas placas y la calidad profesional a la que se encuentra sujeta su diseño requiere de costosas máquinas industriales. Pero no hay por qué desanimarse, ya que actualmente, existe una gran variedad de herramientas o materiales económicos con qué poder realizarlas, solo basta un poco de dinero e ingenio para su construcción.

«La fabricación de una PCB es una tarea muy sencilla, aunque la mayoría de las veces implica tiempo, ingenio, técnica y mucha práctica».

Las placas de circuito impreso incluyen el grabado de pistas o buses por donde transitan los bits (o, en su defecto, el voltaje o la corriente), además de un conjunto de puntos o terminales que representan los pines de aterrizaje de cada componente electrónico. Las PCB pueden o no llevar un tratamiento industrial, lo que implica el cambio de apariencia (color y acabados); aunque, en la actualidad, es muy común contar con elementos caseros que le permiten al usuario obtener un acabo profesional. Actualmente, existe una gran gama de productos de software que permiten el trazado de diagramas para el grabado de placas de circuito impreso. Algunos ejemplos de herramientas de software son: PCB *Wizard/LiveWire*, FreePCB, KiCAD, *CircuitMaker*. Algunas de estas utilerías serán explicadas en el apéndice B de este libro. Más adelante se darán a conocer también los materiales para la construcción de una PCB.

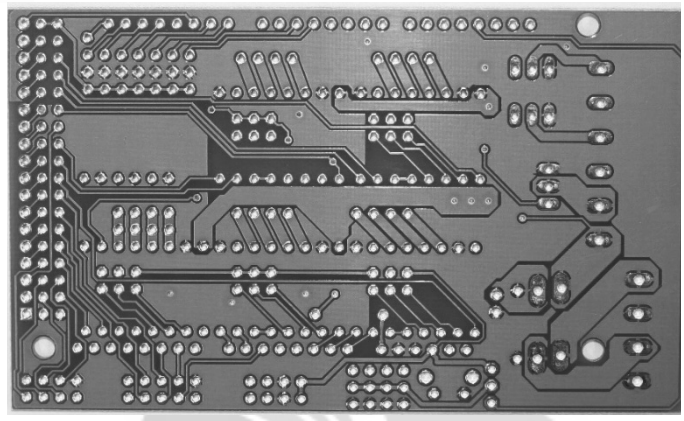


Figura 1.29 PCB integrada por un conjunto de pistas y puntos de aterrizaje

1.6.1 Ingeniería de hardware

La ingeniería de hardware consiste en el estudio de todo lo relacionado con el análisis, diseño, desarrollo, implementación y prueba de un producto que a menudo requiere del uso y aplicación de la electrónica, la física, la ingeniería y la computación. Se sabe que el hardware es todo aquel elemento físico y que «se puede tocar» a diferencia del software que se encuentra comúnmente asociado a una computadora.

La ingeniería de hardware describe tanto las normas como los procedimientos para la fabricación de componentes, artefactos y dispositivos capaces de dotar cualquier sistema (informático) funcional.

La ingeniería de hardware abarca desde el estudio de las unidades de medida de transferencia y almacenamiento de datos, hasta la fabricación y prueba de placas de circuito impreso.

1.6.2 Desarrollo de hardware para PC

Actualmente, se desarrolla una gran variedad de hardware para PC (periféricos), y que, generalmente, se encuentra clasificado en dispositivos de entrada, salida, almacenamiento, procesamiento y comunicación. Todos estos fabricados, en la actualidad, por compañías especializadas en el diseño de hardware para computadoras. La fabricación de hardware se efectúa gracias a la existencia de máquinas o robots industriales (generalmente de origen alemán) localizados en empresas trasnacionales de gran renombre, muchas de ellas de origen asiático o estadounidense.

Para el desarrollo de hardware se necesitan conocimientos sobre precisión matemática, cálculo, física, electrónica, informática (programación), sistemas de control, nanotecnología.

El hardware de una PC puede ser interno o externo. Algunos elementos internos de hardware para PC son: la *motherboard*, las tarjetas periféricas, el microprocesador y los módulos de memoria RAM. Ahora, resulta importante saber que la mayoría de estos elementos son grabados sobre placas de material especial como resinas de fibra de vidrio reforzada (la más conocida es la FR4), cerámica, plástico, teflón y polímeros (como la baquelita), que después son intervenidas por máquinas de montaje para ensamblar componentes (uso de la tecnología de montaje superficial o SMT). El resto del proceso se lista en el apéndice B de este libro.

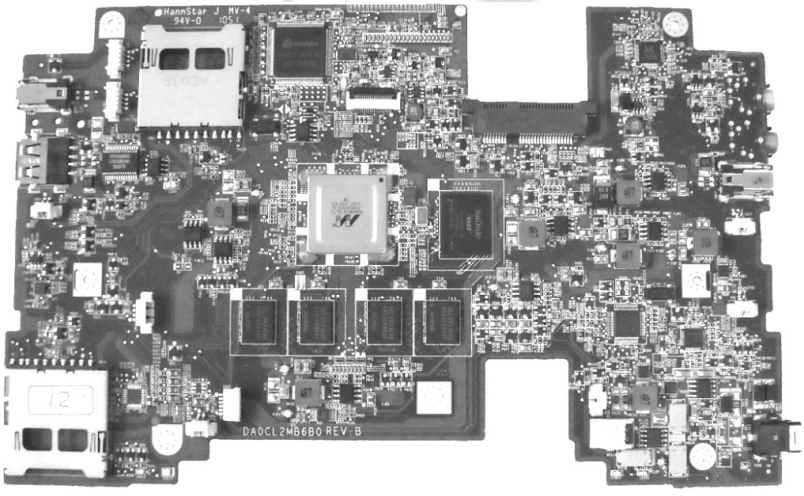


Figura 1.30 La *motherboard* de una PC

ACTIVIDAD 8

1. Desarrolle una aplicación en App Inventor que permita controlar en el encendido y apagado de un foco, asimismo la activación de un motor y un *buzzer*.
2. Trace el diseño de un contador de 2 dígitos 00 a 99 desde LiveWire y genere su esquema PCB.
3. Realice un cuadro comparativo en el que mencione las características de, por lo menos, tres herramientas de software para la generación de una PCB.

En este capítulo se ha analizado diversos conceptos básicos y algunos aspectos que deben considerarse de manera previa para incursionar en el desarrollo de proyectos o aplicaciones electrónicas de hardware. Se ha hecho también una revisión general sobre plataformas de hardware libre de mayor fama en el mercado, citando para ello herramientas para la programación de módulos y lenguajes de propósito específico. En general, se ha revisado la aplicación tanto de la electrónica como de la ingeniería de hardware en la vida cotidiana, proponiendo, luego, ejemplos para su comprensión. En el siguiente capítulo se darán a conocer los sistemas de numeración más comunes y algunas técnicas para la resolución de problemas de índole electrónico.

Capítulo 2

Sistemas numéricos

EDITORIAL
MACRO®

El estudio de los sistemas numéricos es de suma importancia, sobre todo a la hora de comenzar a analizar, desarrollar y programar hardware. Es necesario, por lo tanto, conocer desde las unidades de medida necesarias para la realización de cálculos hasta el proceso lógico que llevan a cabo los sistemas electrónicos.

Dentro del contexto de los sistemas numéricos se pueden ubicar tópicos como métodos para la construcción de números, conversiones numéricas de distintas bases (2, 8, 10, 16), operaciones aritméticas básicas entre bits y bytes, manejo de caracteres y su representación (código ASCII), operaciones que involucran el uso de frecuencias y cálculo de transferencias, etc.

2.1 UNIDADES DE MEDIDA DE ALMACENAMIENTO Y TRANSFERENCIA DE DATOS

En términos informáticos, existen por lo general dos unidades de medida básicas para cualquier cálculo. Se trata de la unidad de medida de transferencia y la unidad de medida de almacenamiento de datos.

La **unidad de medida de transferencia** de datos es el **bit**. Esta unidad es comúnmente utilizada para expresar el ancho de banda presente en medios físicos de conexión como cables, puertos o interfaces presentes en equipos de cómputo o de red. Además, sirven para representar señales digitales y el ancho de datos en algunos dispositivos.

Ejemplo de aplicación de los bits en la vida moderna

- Para expresar el ancho de banda en un dispositivo de red, un *switch* contiene 8 puertos RJ-45, los cuales trabajan a 100 Mbps. Esta cifra se lee como 100 megabits por segundo.
- Para conformar direcciones IP, la dirección IP asignada al puerto serial 0/0/0 de un *router* es: 192.168.2.9, la cual se lee 11000000.10101000.00000010.00001001.
- Obtener la capacidad de transferencia de dispositivos, un microprocesador Intel Core i3 de 2,27GHz, trabaja con un ancho de datos de 32 bits.

La palabra «bit» es un término compuesto (*binary digit*) que puede representarse por un 0 o 1. Originalmente, se concibe como la unidad mínima de un dato. Un 0 binario representa la ausencia de voltaje, mientras que el 1 binario alude a la presencia de 5 Volts.

La **unidad de medida de almacenamiento de datos** es el **byte**. A menudo, los dispositivos de almacenamiento, como el disco duro o las memorias USB, hacen uso de este tipo de unidades para expresar su capacidad. «Byte» proviene de las palabras *binary multe* (multiplicación binaria).

A menudo los dispositivos de almacenamiento, por ejemplo, el disco duro o las memorias USB hacen uso de este tipo de unidades para expresar su capacidad.

ACTIVIDAD 1

Examine el administrador de equipos de su propia computadora y obtenga los datos de capacidad de almacenamiento de las unidades instaladas:

a) C:/>: _____

b) Unidad de DVD: _____

c) Disco extraíble: _____

Por otro lado, el «byte» es un término compuesto por las palabras *binary multe* o múltiplo binario. Se conoce también con el nombre de octeto, pues se encuentra formado por ocho bits consecutivos. La mitad de un byte se llama *nibble* (conjunto de 4 bits).

Figura 2.1 Disco duro, unidad de hardware cuya capacidad de almacenamiento se expresa en bytes

2.2 MULTIPLICADORES BINARIOS

Los multiplicadores también son conocidos como prefijos. Estos, a menudo, son utilizados para expresar una multiplicación de unidades. En el ámbito computacional se utilizan para expresar frecuencias (f) y capacidades de transferencia y almacenamiento, aunque, se debe saber que los valores dependerán del sistema utilizado (decimal o binario). Originalmente, los prefijos tienen un equivalente estándar el cual se muestra en la tabla:

Tabla 2.1 *Valor de los prefijos o multiplicadores de una unidad de medida*

Símbolo	Nombre	Equivalencia	Significado
K	Kilo	1000	10^3
M	Mega	1 000 000	10^6
G	Giga	1 000 000 000	10^9
T	Tera	1,000 000 000 000	10^12
P	Peta	1 000 000 000 000 000	10^15
E	Exa	1 000 000 000 000 000 000	10^18
Z	Zetta	1 000 000 000 000 000 000 000	10^21
Y	Yotta	1 000 000 000 000 000 000 000 000	10^24

Por otro lado, se tiene el manejo de prefijos exclusivos para representaciones binarias (almacenamiento y transferencia de datos).

Tabla 2.2 *Valor de los prefijos o multiplicadores binarios*

Símbolo	Nombre	Equivalencia	Significado	Valor
K	Kibi	1024	2^10	1024 Bytes
M	Mebi	1 048 576	2^20	1024 KB
G	Gibi	1 073 741 824	2^30	1024 MB
T	Tebi	1 099 511 627 776	2^40	1024 GB
P	Pebi	1 125 899 906 842 624	2^50	1024 TB
E	Exbi	1 152 921 504 606 846 976	2^60	1024 PB
Z	Zebi	1 180 591 620 717 411 303 424	2^70	1024 EB
Y	Yobi	1 208 925 819 614 629 174 706 176	2^80	1024 ZB

2.3 FÓRMULAS Y MÉTODOS PARA LA CONSTRUCCIÓN DE NÚMEROS Y CONVERSIONES

Actualmente, son muchos los métodos o técnicas para la construcción de cantidades numéricas, sobre todo si se trata de representaciones decimales o binarias. Una forma sencilla, pero formal, para la construcción de números (inclusive para efectuar conversiones numéricas entre distintos sistemas) consta de utilizar la siguiente fórmula general:

$$N = \sum_{i=-1}^{i=0} a * R^{i}$$

Donde:

N = Cualquier número susceptible a construcción (binario, decimal, otro)
(S) = Sumatoria de cada dígito que conforma a N
a = Coeficiente – dígito que conforma a N
R = Base o raíz
i = Subíndice, el cual determina la posición o potencia para R

Ejemplo de posicionamiento

$$N = \underbrace{4^{2}\ 5^{1}\ 6^{0}}\ ,\ \underbrace{4^{-1}\ 5^{-2}}$$

Estas expresiones (datos) pueden ser de tipo real o flotante.

Ejemplo de aplicación de la fórmula general para cantidades decimales

230d = 2(10) ^ 2+ 3(10) ^ 1 + 0(10) ^ 0
230d = 2(100) + 3(10) + 0(1)
230d = 200 + 30 + 0
230d = 230d

Esta fórmula puede también ser aplicable para conversiones numéricas de distinta base.

Ejemplo de aplicación de la fórmula general para cantidades binarias

1010b = 1(2) ^ 3 + 0(2) ^ 2 + 1(2) ^ 1 + 0(2) ^ 0
1010b = 1(8) + 0(4) + 1(2) + 0(1)
1010b = 8 + 0 + 2 + 0
1010b = 10d

ACTIVIDAD 2

Realice los siguientes ejercicios aplicando la fórmula general para la construcción de un número. Convertir una cantidad expresada en base 2, 8 o base 16 (según sea el caso) a una cantidad en base 10.

a) 10010110b: _____
b) 10110010b: _____
c) 00011111b: _____
d) 11100011b: _____
e) 11110001b: _____
f) 28o: _____
g) 111o: _____
h) 777o: _____
i) 1001h: _____
j) 45h: _____

El sistema binario también es conocido como sistema posicional. El origen de este, se puede explicar con la siguiente tabla de posiciones y valores:

Tabla 2.3 *Posición de los valores del sistema binario (regla posicional)*

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

ACTIVIDAD 3

1. Obtenga la solución de las siguientes operaciones (potencias).

   a) 2^7: _____
   b) 2^10: _____
   c) 2^8: _____
   d) 2^11: _____
   e) 2^9: _____
   f) 2^12: _____

2. Obtenga el resultado en decimal de las siguientes cantidades. Para ello, utilice el sistema posicional (desglose su procedimiento).

   a) 11100011b: _____
   b) 10111011b: _____
   c) 10110110b: _____
   d) 11110110b: _____

Antes de abordar el tema de conversiones numéricas, es preciso comenzar a hablar sobre los sistemas más empleados en la actualidad. A continuación, se listan dichos sistemas y algunas recomendaciones:

Nota: Con la finalidad de comprender el presente tema, se ha planteado un orden estratégico para ir revisando uno a uno los sistemas numéricos más representativos.

El orden propuesto es el siguiente:

- Sistema binario.
- Sistema decimal.
- Sistema octal.
- Sistema hexadecimal.

Hay que destacar que para poder trabajar con cada uno de estos sistemas, se debe partir del manejo de bytes como unidad de medida límite para la obtención de expresiones; es decir, mantener un arreglo de ocho bits por cada representación y su respectivo resultado. Ejemplos de bytes: 10001000, 11110001, 10101010.

Para la identificación de cualquier sistema de numeración basta con colocar un sufijo o subíndice en cada expresión. Por ejemplo, un valor expresado en decimal se puede colocar del siguiente modo: 123d o, simplemente, 123(10). En caso de utilizar un sistema octal o hexadecimal, quedaría de la siguiente manera: 514o/514(8) y 4Dh/4D(16), respectivamente.

2.3.1 Sistema binario y decimal

Por un lado se tiene el sistema binario, el cual es considerado como el más simple de todos. Maneja dos posibles valores: el 0 lógico y el 1 lógico. Se trata del sistema utilizado por los ordenadores para su computación, asimismo representa la base para el manejo de la mayoría de los sistemas numéricos. El sistema binario, o de base dos, se emplea mucho para el diseño e implementación de circuitos digitales. Para la manipulación de compuertas, a menudo, se emplean valores como el 0 y el 1 lógico, ya que gracias a estos datos se pueden construir **tablas de verdad** y **mapas de Karnaugh**.

La aplicación de los valores binarios en la electrónica se concibe mediante el uso de interruptores o ledes. Lo anterior quiere decir que, para poder representar las entradas de un circuito electrónico (niveles de tensión válidos en la electrónica digital), se utilizan los *switches* o interruptores; en tanto que, para poder visualizar los posibles estados de salida en un circuito se utilizan los ledes.

Por otro lado, se tiene el sistema decimal que es el más utilizado en la actualidad; comprende valores del 0 al 9, los cuales pueden ser manipulados desde un conjunto de interruptores o pulsadores y visualizados a través de ledes o *displays* de siete segmentos. La combinación de esos dígitos ofrece como resultado un valor infinito (n).

Tabla 2.3 *Valores binarios y su equivalencia en decimal*

Valor en binario (byte)	Valor en decimal
00000000	0
00000001	1
00000010	2
00000011	3
00000100	4
00000101	5
00000110	6
00000111	7
00001000	8
00001001	9
N	n

2.3.2 Sistema octal

El sistema octal se conoce también como sistema de base 8. Por lo general, deben emplearse tres bits para la representación de sus posibles valores (que van de 0 al 7). Es necesario saber que dado un byte, se pueden expresar hasta dos dígitos en octal; pero si se desea tener tres dígitos, es necesario agregar un bit a la izquierda para su validación (completar con 0 lógico). Por ejemplo, (0)10 100 111 = 247o. En la siguiente tabla se muestra una equivalencia entre el sistema decimal y octal:

Tabla 2.4 *Valores octales y su equivalencia en binario*

Valores decimales	Valores en octal	Valor en binario (byte)
0	0	00 000 000
1	1	00 000 001
2	2	00 000 010
3	3	00 000 011
4	4	00 000 100
5	5	00 000 101
6	6	00 000 110
7	7	00 000 111
8	No válido	No aplica
9	No válido	No aplica

2.3.3 Sistema hexadecimal

El sistema hexadecimal es también conocido como sistema de base 16, ya que maneja valores de entre 0 a 15. Este sistema, a diferencia del resto, emplea letras (A, B, C, D, E, F) que cumplen con sustituir algunos valores numéricos. Para comprender mejor esta parte, se propone la siguiente tabla:

Tabla 2.5 *Valores hexadecimales y su equivalencia en binario*

Valores decimales	Valores en hexadecimal	Valor en binario (dos *nibbles*)
0	0	0000 0000
1	1	0000 0001
2	2	0000 0010
3	3	0000 0011
4	4	0000 0100
5	5	0000 0101
6	6	0000 0110
7	7	0000 0111
8	8	0000 1000
9	9	0000 1001
10	A	0000 1010
11	B	0000 1011
12	C	0000 1100
13	D	0000 1101
14	E	0000 1110
15	F	0000 1111

Para la representación del sistema hexadecimal a menudo se emplea el *nibble* (conjunto de 4 bits). Aunque dentro de un byte se pueden manejar hasta dos dígitos en hexadecimal. Por ejemplo: 1110 1100 = ECh.

Con el propósito de seguir conociendo más sobre la electrónica y la ingeniería de hardware, es necesario atender algunos aspectos relacionados con el ámbito de las matemáticas computacionales. Para ello, es necesario atender los tipos de conversiones más representativas:

- N(x) → N(10)
- N(10) → N(x)
- N(2) → N(8)
- N(2) → N(16)

Donde:

x = Puede tomar valores de un sistema en binario, decimal, octal o hexadecimal

10 = Expresiones decimales

2 = Expresiones binarias

8 = Expresiones octales

16 = Expresiones hexadecimales

2.3.4 Aplicación de la fórmula general para la construcción de números

Hay que recordar que la fórmula general, para la construcción de números, tiene dos funciones principales: la primera, es ayudar al usuario a encontrar la forma en que se construye un número mediante el empleo de unidades, decenas y centenas; la segunda, es para efectuar ciertas conversiones numéricas. La fórmula general es válida para los siguientes casos:

- De binario a decimal.
- De decimal a decimal.
- De octal de decimal.
- De hexadecimal a decimal.

.

Ejemplo de resolución de conversiones numéricas mediante el uso de la fórmula general para la construcción de números

a. Convierta el valor 135o (expresado en octal) a base 10 (valor en decimal).

135o = 1(8)^2 + 3(8) ^1 + 5(8) ^0

135o = 1(64) + 3(8) + 1(5)

135o = 64 + 24 + 5

135o = 93d

b. Convierta el valor 8Dh (expresado en hexadecimal) a base 10 (valor en decimal).

8Dh = 8(16) ^1 + 13(16) ^ 0

8Dh = 8(16) + 1(13)

8Dh = 128 + 13

8Dh = 141d

Para comprobar los resultados se pueden utilizar métodos alternos (los cuales se describirán más adelante), aunque existe la posibilidad de hacerlo desde la PC. Para ello, es necesario abrir la calculadora de Windows (o, en su defecto, utilizar el sistema operativo empleado por el usuario), en modo programador.

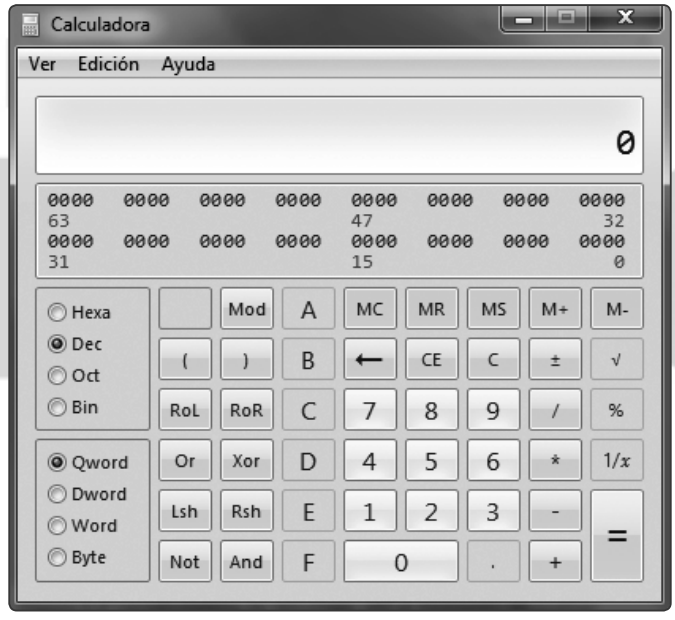


Figura 2.2 La calculadora de Windows, útil al momento de realizar conversiones de distintas bases

Existen diferentes métodos o técnicas para realizar conversiones numéricas de distintas bases, por ejemplo el método de la escalera para obtener residuos de una división (conocido también como método de la división por la base). Esta técnica es utilizada, a menudo, para conversiones de decimal a binario, decimal a octal y decimal a hexadecimal. Lo único que se tiene que hacer es dividir la cantidad expresada en decimal entre 2, 8, 16 según sea el caso.

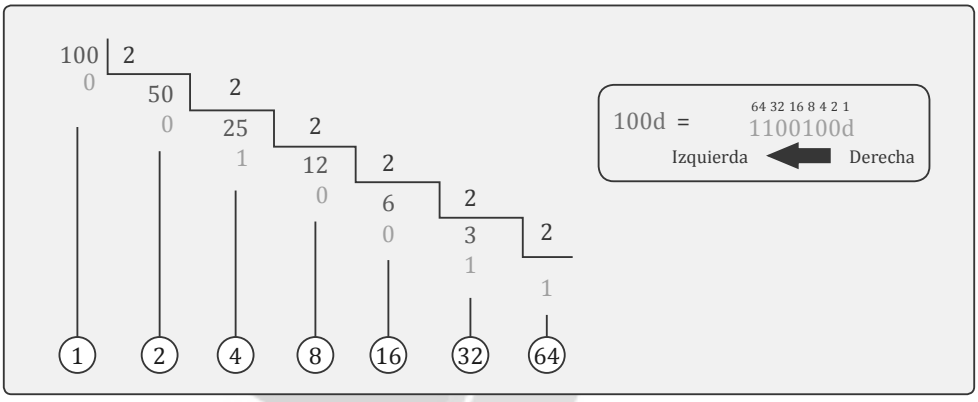


Figura 2.3 Método de la escalera para conversiones de base 10 a base 2,8 o 16
Fuente: el autor.

Para la obtención de un resultado idóneo, es necesario saber acomodar los bits. Para ello, es aconsejable hacerlo de arriba hacia abajo en el orden establecido por la regla posicional (derecha a izquierda) como se aprecia en la figura anterior. El procedimiento para convertir un número decimal a otra base es el mismo. Para comprobar el resultado obtenido, se puede emplear la regla posicional (tabla 2.3).

2.4 OPERACIONES ARITMÉTICAS DE BASE DOS

Se sabe que las operaciones aritméticas básicas son la suma, la resta, la multiplicación y la división. En este apartado se desarrollan dos de las cuatro operaciones aritméticas, la suma y la resta que son las más utilizadas en el ámbito de la electrónica digital y la ingeniería de hardware.

En un principio, estas operaciones son de bastante ayuda para comprender el funcionamiento del hardware de cualquier sistema; por lo mismo, es importante que las expresiones binarias que se desean sumar, restar, multiplicar o dividir estén conformadas en paquetes de ocho bits (byte), esto con la finalidad de hacer más fácil su lectura e interpretación[1].

[1] En *Matemáticas para programadores* de Barden, Jr. (1986), se proporciona un estudio minucioso sobre los sistemas numéricos y la aritmética binaria.

2.4.1 La suma binaria

La suma binaria sigue el mismo principio que la suma decimal, solo que en este caso han de manejarse solamente el 0 lógico y el 1 lógico. Las partes de la suma y el procedimiento para efectuarla son los mismos para ambos sistemas numéricos. Para hacer más sencilla su solución, se plantea la siguiente tabla de estados:

Tabla 2.6 *Tabla de estados de la suma binaria*

S_1		S_2		Res
0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	0 y se acarrea 1

La tabla de estados de la suma binaria representa una guía para partir con su desarrollo. En esta se muestran tres campos: S_1 y S_2 que hacen referencia a sumando 1 y 2 respectivamente y el campo Res que muestra el resultado obtenido.

Antes de comenzar a plantear ejemplos sobre suma binaria, resulta importante dar un repaso a las partes de la suma: sumando_1, sumando_2, hasta sumando_n y la suma o resultado. Estos elementos se muestran en la siguiente tabla con un ejemplo:

Tabla 2.7 *Ejemplo de suma binaria de ocho bits*

Byte(s)	<<	128	64	32	16	8	4	2	1
Sumando_1		1	0	1	0	1	0	1	0
Sumando_2		1	0	1	0	1	0	1	0
	+								
Suma	1	0	1	0	1	0	1	0	0

El procedimiento para efectuar una suma binaria de ocho bits es el siguiente:

1. Partiendo de una suma de ocho bits con dos sumandos. En primer lugar, debe identificar que ambos sumandos tengan un formato válido de ocho bits. De no ser así, se recomienda agregar ceros a la izquierda (no afectan la operación, ni interfieren con el valor proporcionado) para conformar el byte.

Posición		128	64	32	16	8	4	2	1
Sumando_1		1	0	0	1	1	0	1	1
Sumando_2	+	0	0	0	1	1	0	1	1
Resultado									

2. Con base a la tabla de estados de la suma, comenzar la operación de la derecha a la izquierda y realizar la suma del primer sumando con el segundo sumando. Nótese que al efectuar la operación 1 + 1, se tiene como resultado un 2, el cual se expresa como 10b. Si toma la regla posicional de derecha a izquierda, se obtiene la equivalencia del 2 binario. Se procede, entonces, a colocar el 0 lógico en la primera posición del resultado iniciando de derecha a izquierda y acarreando un 1 lógico por encima del primer sumando.

Acarreo								1	
Sumando_1		1	0	0	1	1	0	1	1
Sumando_2	+	0	0	0	1	1	0	1	1
Resultado		X	x	x	x	x	x	x	0

3. Realizar lo anterior para las posiciones posteriores.

Posición		128	64	32	16	8	4	2	1
Acarreo				1	1		1	1	
Sumando_1		1	0	0	1	1	0	1	1
Sumando_2	+	0	0	0	1	1	0	1	1
Resultado		1	0	1	1	0	1	1	0

4. El resultado de la suma se puede comprobar utilizando algún método alterno como la regla posicional, o mediante la conversión de base 2 a base 10.

Posición		128	64	32	16	8	4	2	1	DEC
Acarreo				1	1		1	1		
Sumando_1		1	0	0	1	1	0	1	1	155
Sumando_2	+	0	0	0	1	1	0	1	1	27
Resultado		1	0	1	1	0	1	1	0	182

ACTIVIDAD 4

1. Realice las siguientes sumas binarias:
   - **a)** 11101100 + 11010111 = _____
   - **b)** 01010101 + 00000111 = _____
   - **c)** 00111010 + 11000000 = _____
   - **d)** 11010001 + 11101010 = _____
   - **e)** 00100011 + 11110110 = _____

2. Convierta los siguientes valores de base 10 a base 2 y efectúe la operación señalada:
   - **a)** 11 + 250 = _____ + _____ = _____
   - **b)** 63 + 32 = _____ + _____ = _____
   - **c)** 100 + 17 = _____ + _____ = _____
   - **d)** 45 + 152 = _____ + _____ = _____
   - **e)** 15 + 15 = _____ + _____ = _____

2.4.2 La resta binaria

La resta binaria incluye un parámetro llamado préstamo, a menudo llamado acarreo-sustracción. Para hacer más sencillo el proceso de resolución de una resta binaria, se propone una tabla de estados similar a la utilizada para la suma:

Tabla 2.8 *Tabla estados de la resta binaria*

M		S		Res
0	-	0	=	0
0	-	1	=	1 préstamo 1
1	-	0	=	1
1	-	1	=	0

La tabla de estados de la resta binaria representa una guía para partir de su desarrollo; aunque, en este caso no se cuenta con un acarreo, sino con un préstamo a diferencia de la suma binaria. En esta se muestran tres campos: M y S que hacen referencia al minuendo y sustraendo, respectivamente, el campo Res solo muestra el resultado obtenido.

Antes de comenzar a plantear ejemplos sobre resta binaria, resulta importante dar un repaso a las partes de la resta, al primer registro se le llama minuendo; al segundo registro, sustraendo; y al resultado, resta o diferencia. Para la obtención de resultados positivos, es necesario que el minuendo (M) sea mayor que el sustraendo (S).

Los elementos antes mencionados se muestran en la siguiente tabla:

Tabla 2.9 *Ejemplo de resta binaria de ocho bits*

Byte(s)	<<	128	64	32	16	8	4	2	1
Minuendo		1	1	1	0	0	0	1	0
Sustraendo		0	0	0	0	1	0	1	0
	-								
Diferencia		1	1	0	1	1	0	0	0

El procedimiento para efectuar una resta binaria de ocho bits es el siguiente:

1. Partiendo de una resta de ocho bits, primero debe identificar que tanto el minuendo como el sustraendo tengan un formato válido de ocho bits. De no ser así, se recomienda agregar ceros a la izquierda (no afectan la operación, ni interfieren con el valor proporcionado) para conformar el byte.

Posición		128	64	32	16	8	4	2	1
Minuendo		1	1	0	0	0	0	0	0
Sustraendo	-	0	0	0	1	0	0	0	1
Diferencia									

2. Con base a la tabla de estados de la resta, comenzar la operación de derecha a izquierda y realizar la resta correspondiente. Nótese que al efectuar la operación 0 + 1, se tiene como resultado un 1. Lo que implica tener que solicitar un 1 lógico al bit inmediato siguiente. Este bit se cobra hasta el momento de encontrarse con un 1 lógico.

Préstamo			1	1	1	1	1	1	
Minuendo		1	1	0	0	0	0	0	0
Sustraendo	-	0	0	0	1	0	0	0	1
Diferencia		1	0	1	0	1	1	1	1

3. Realizar lo anterior para las posiciones posteriores.

Posición		128	64	32	16	8	4	2	1
Préstamo			1	1	1	1	1	1	
Minuendo		1	1	0	0	0	0	0	0
Sustraendo	-	0	0	0	1	0	0	0	1
Diferencia		1	0	1	0	1	1	1	1

4. El resultado de la resta se puede comprobar utilizando algún método alterno como la regla posicional, o mediante la conversión de base 2 a base 10.

Posición		128	64	32	16	8	4	2	1	DEC
Préstamo			1	1	1	1	1	1		
Minuendo		1	1	0	0	0	0	0	0	192
Sustraendo	-	0	0	0	1	0	0	0	1	17
Diferencia		1	0	1	0	1	1	1	1	175

Figura 2.4 Binary Game, utilería *online* que permite interactuar con valores binarios

ACTIVIDAD 5

1. Realice las siguientes restas binarias:

   a) 10101111 – 00011110 = _____
   b) 11010100 – 00001011 = _____
   c) 11110100 – 00000010 = _____
   d) 11010000 – 00000101 = _____
   e) 00110011 – 00001010 = _____

2. Convierta los siguientes valores de base 10 a base 2 y efectúe la operación señalada:

   a) 200 – 100 = _____ – _____ = _____
   b) 63 – 32 = _____ – _____ = _____
   c) 2000 – 325 = _____ – _____ = _____
   d) 45 – 20 = _____ – _____ = _____
   e) 119 – 95 = _____ – _____ = _____

Práctica de laboratorio n.º 1

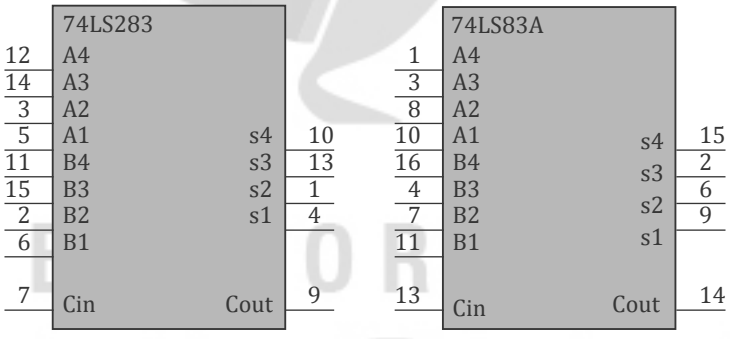
El sumador binario

El CI 74283 hace la suma de dos números de cuatro bits, A3 A2 A1 A0 y B3 B2 B1 B0, más un acarreo de entrada C0 (Cin) que proviene de una suma previa:

$$\begin{array}{rr} & C_0 \\ & A_3\ A_2\ A_1\ A_0 \\ + & B_3\ B_2\ B_1\ B_0 \\ \hline \end{array}$$

Estas entradas se proporcionan manualmente +5 V para 1, 0 V para un 0

En la figura se muestra el circuito lógico de un circuito sumador. Al no indicar el número de cada terminal, los dos dispositivos se ven idénticos. Por esta razón, es importante identificar el número del componente y sus terminales. Los pines de VCC y Tierra no aparecen en el diagrama de esta figura, por omisión todos los circuitos ya están polarizados a 5 V.



Distribución de terminales de los CI 74283 y 7483

Fundamento teórico:

Las entradas que quedan flotando (esto es, sin conectar) son interpretadas por el CI como unos. Si no se desea un acarreo hacia la primera columna (C_0=0), entonces debe conectarse a tierra.

Las salidas son el resultado de la suma, y se conectan a los ledes para observarlas. Un 1 en la salida debe encender el led, mientras que un 0 no debe hacerlo. El C0 es un acarreo que viene de una suma previa. Los acarreos C1, C2 y C3 los maneja el CI internamente, y C4 (Cout) representa el sobreflujo o acarreo hacia la columna siguiente.

Las otras salidas se marcan con ∑ (sigma) y denotan la suma $\sum_3$, $\sum_2$, $\sum_1$, $\sum_0$ y representan las sumas de las columnas 4, 3, 2, 1, respectivamente.

$$\begin{array}{rcccc} & & & & C_0 \\ & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ \hline C_4 & \sum_3 & \sum_2 & \sum_1 & \sum_0 \end{array}$$

El **CI 74283** está diseñado para realizar la operación de suma A+B. Si se requiere de otra operación como la resta de A-B, entonces el valor de B debe ser negativo. El valor negativo de B se puede representar en complemento a 1 o complemento a 2.

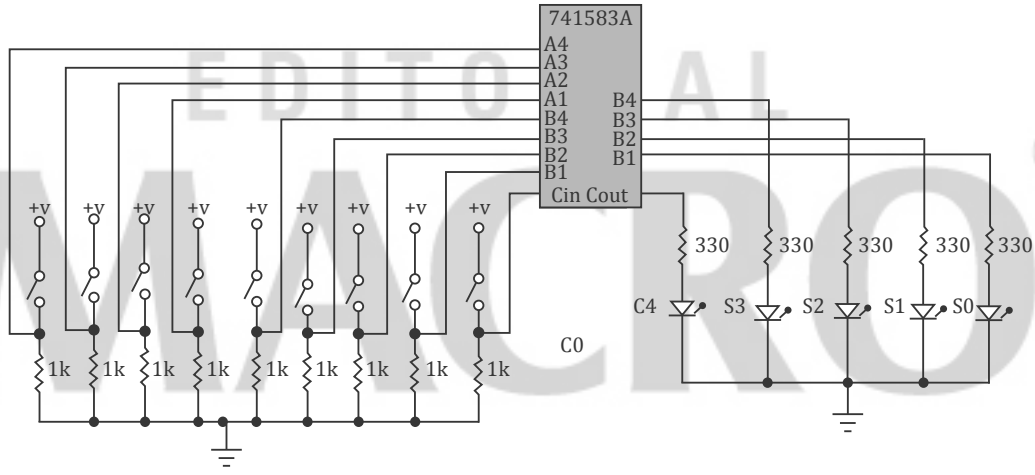
Procedimiento:

1. Conecte los componentes como se muestra en la figura (realizarlo sobre algún simulador y de manera física).
2. Investigue el método para resolver una suma y una resta por complemento a 1 y 2.

_____

_____

_____



Sumador de 4 bits

3. Utilizando el circuito de la figura anterior, realice cinco sumas de números positivos. Sobre la misma tabla escriba, en el formato binario, los números que se van a sumar y el resultado.

A	B	Sigma	A (binario)	B (binario)	Resultado (binario)
8	7				
9	5				
10	6				
3	1				
7	3				

Cuestionario final

1. ¿Cuál es la diferencia entre un circuito integrado sumador 7483 y un 74283?

_____

_____

_____

2. ¿Cómo se puede realizar una resta, utilizando un circuito sumador?

_____

_____

_____

3. ¿Qué sucede si la terminal del acarreo de entrada Cin (C0) no se conecta ni a tierra ni VCC?

_____

_____

_____

2.5 EL CÓDIGO ASCII

El código ASCII (*American Standard Code Information Interchange*) consiste en una tabla de caracteres basada en el alfabeto latino. Contiene letras, números, signos-símbolos, etc. Se usa básicamente como guía para entender la estructura de un teclado de computadora convencional. Cada carácter se compone de ocho bits. Las tabla de código ASCII, a menudo, integra expresiones decimales, binarias, hexadecimales.

«El código ASCII se divide en tres secciones: caracteres de control (0-31), caracteres imprimibles o código estándar (32-128) y el código extendido (128-255)» (Acha, Castro, Pérez, Rioseras, 2006: p. 83).

Como se ha mencionado, su estudio representa el principio del diseño, integración e implementación de teclado. El teclado matricial, por ejemplo, es comúnmente utilizado en sistemas de seguridad como checadores de entrada y salida, lectores de código, alarmas, cajas fuerte, ya que utilizan un panel numérico y en ciertas ocasiones algunas letras del abecedario. Existen también teclados alfanuméricos (QWERTY) para proyectos electrónicos o PC.

Tabla 2.10 *Esquema que ilustra un fragmento del código ASCII*

Caracteres no imprimibles			Caracteres imprimibles					
Nombre	Dec	Car.	Dec	Car.	Dec	Car.	Dec	Car.
Nulo	0	NUL	32	Espacio	64	@	96	`
Inicio de cabecera	1	SOH	33	!	65	A	97	a
Inicio de texto	2	STX	34	“	66	B	98	b
Fin de texto	3	ETX	35	#	67	C	99	c
Fin de transmisión	4	EOT	36	$	68	D	100	d
enquiry	5	ENQ	37	%	69	E	101	e
acknowledge	6	ACK	38	&	70	F	102	f
campanilla (beep)	7	BEL	39	‘	71	G	103	g
backspace	8	BS	40	(	72	H	104	h
Tabulador horizontal	9	HT	41	)	73	I	105	i
Salto de línea	10	LF	42	*	74	J	106	j
Tabulador vertical	11	VT	43	+	75	K	107	k
Salto de página	12	FF	44	,	76	L	108	l
Retorno de carro	13	CR	45	-	77	M	109	m
Shift fuera	14	SO	46	.	78	N	110	n
Shift dentro	15	SI	47	/	79	O	111	o
Escape línea de datos	16	DLE	48	0	80	P	112	p
Control dispositivo 1	17	DC1	49	1	81	Q	113	q
Control dispositivo 2	18	DC2	50	2	82	R	114	r
Control dispositivo 3	19	DC3	51	3	83	S	115	s

Control dispositivo 4	20	DC4	52	4	84	T	116	t
Neg acknowledge	21	NAK	53	5	85	U	117	u
Sincronismo	22	SYN	54	6	86	V	118	v
Fin bloque transmitido	23	ETB	55	7	87	W	119	w
Cancelar	24	CAN	56	8	88	X	120	x
Fin medio	25	EM	57	9	89	Y	121	y
Sustituto	26	SUB	58	:	90	Z	122	z
Escape	27	ESC	59	;	91	[	123	{
Separador archivos	28	FS	60	<	92	\	124	|
Separador grupos	29	GS	61	=	93	]	125	}
Separador registros	30	RS	62	>	94	^	126	~
Separador unidades	31	US	63	?	95	_	127	DEL

Un modo habitual para el despliegue de caracteres, desde cualquier teclado para PC, se consigue pulsando la tecla **Alt** + tecla del carácter. Algunas teclas tienen asignadas por *default* el carácter que la identifica.

Ejemplo de uso del código ASCII

El número 32 representa un espacio.

Del número 48 hasta el 57 son los números naturales desde el 0 hasta el 9.

Del 65 hasta el 90 son letras de A hasta la Z en mayúsculas.

De 97 hasta 122 son letras de a hasta la z en minúsculas.

Ejemplo:

Alt + 1 = ☺

Alt + 2 = ☻

Alt + 64 = @

Alt + 100 = d

ACTIVIDAD 6

1. Complete el siguiente cuadro haciendo uso de la tabla del código ASCII:

Decimal	Binario (Byte)	ASCII
112	-	
41	-	
	-	u
	-	9
	00000111	
32		
205		
		X
	11010010	
56	00111000	

2. Use la tabla del código ASCII para traducir lo siguiente, según corresponda:

a) 5349204e4f20444553434946524153204553544204d454e53414a 452c2059 412045535441532052455 0524f4241444f (traducir de hexadecimal a carácter ASCII).

_____

b) El amor en tiempos de cólera (traducir a código hexadecimal).

_____

2.6 CÁLCULOS DE CAPACIDAD DE TRANSFERENCIA

Se define como capacidad de transferencia de datos a la cantidad de información que se puede transmitir desde un puerto origen (de cualquier dispositivo) hacia un destino, considerando como vía de transporte al bus de sistema.

Un bus es definido como una carretera por donde viajan los datos. La capacidad de transferencia de un dispositivo cualquiera estará siempre expresada en MB/s.

En el ámbito de la ingeniería de hardware, frecuentemente, se utilizan fórmulas matemáticas para el cálculo de estas capacidades. Para ello, es necesario conocer algunos datos como frecuencia del dispositivo (unidad de medida expresada como hercio) y el ancho de datos (o tamaño de datos) con el que trabaja (expresado en bits).

Para entender lo anterior, analice lo siguiente:

«El **hercio** es la unidad de medida de **frecuencia** (número de veces que suscita algún evento respecto al tiempo), se conoce como hertz y se expresa con las letras Hz» (González, 2012: p. 39). Como toda unidad de medida, y con la ayuda de un prefijo, se establece un conjunto de valores:

1 Hz → 1 ciclo de reloj
1KHz → 1,000 Hz
1 MHz → 1, 000, 000 Hz
1 GHz → 1,000, 000,000 Hz

ACTIVIDAD 7

1. Dados los siguientes valores, conviértalos a ciclos de reloj (Hz).

   a) 2,27 GHz: _____ Hz
   b) 33 MHz: _____ Hz
   c) 66 MHz: _____ Hz
   d) 3.2 GHz: _____ Hz
   e) 4.43 GHz: _____ Hz

Generalmente, los equipos de cómputo integran microprocesadores que trabajan con un ancho de datos de 32 y 64 bits. Los sistemas operativos (SO) no son la excepción, ya que tanto la CPU como el SO trabajan en conjunto para hacer posible el funcionamiento correcto de la PC. Por lo tanto, los SO también trabajan con 32 y 64 bits.

El cálculo de la capacidad de transferencia implica el manejo de una frecuencia y de un ancho de datos. Esta se calcula mediante la siguiente fórmula matemática:

$$f\left(\frac{[P]\text{Hz}}{\text{s}}\right)\left(n\frac{bits}{\text{Hz}}\right)\left(\frac{1\text{Byte}}{8\text{bits}}\right)=\left(CT\frac{\text{MB}}{\text{s}}\right)$$

Donde:
CT = Capacidad de transferencia del dispositivo o medio
f = Frecuencia expresada en Hercios
n = Ancho de datos expresado en bits

Ejemplo de cálculo de la capacidad de transferencia de datos de un dispositivo

Un microprocesador Intel Core i3 de 2,27GHz trabaja con un ancho de datos de 32 bits. Calcular la capacidad de transferencia de datos de dicho dispositivo.

Datos	Operación	Resultado
f= 2,27GHz n = 32 bits CT = ?	CT → (2,27GHz/s) (32bits) (1B/8b)	CT = 9.8GB/s

Con lo anterior se deduce que 32 bits circulan por un bus nada menos que 2 270 000 000 veces en un segundo; y, por lo tanto, la capacidad de transferencia es de 9 800 000 000 bytes por segundo.

ACTIVIDAD 8

1. Calcule la capacidad de transferencia dados los siguientes valores:

   a) 2,3 GHz y 32 bits: _____
   b) 1,8 GHz y 64 bits: _____
   c) 1,4 GHz y 32 bits: _____
   d) 66 MHz y 32 bits: _____
   e) 256 MHz y 64 bit: _____

2. Calcule la capacidad de transferencia dados los siguientes valores:

a) Memoria SD-RAM133	133 MHz	32 bits
b) Memoria RAM266 DDR	133 MHz * 2	32 bits
c) Memoria 400 DDR	200 MHz * 2	64 bits
d) Memoria 800 (400 DDR D Ch)	200 MHz * 2 * 2	64 bits

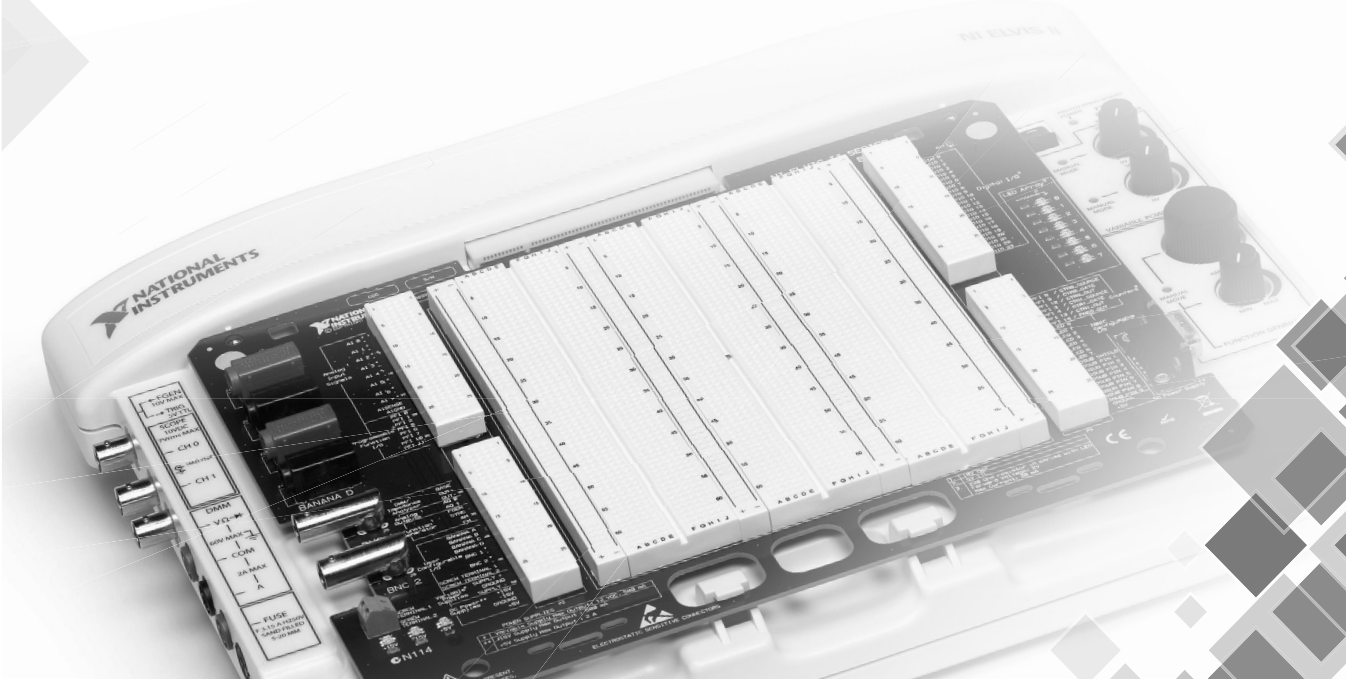
a) _____ c) _____
b) _____ d) _____

En este capítulo se ha analizado los sistemas numéricos más representativos; además, de las conversiones numéricas entre distintas bases más utilizadas. Se han explicado algunas técnicas de desarrollo para dar solución a las operaciones de suma y resta binaria, formulando actividades y prácticas para su comprensión. Al final se revisaron temas como el código ASCII y el cálculo de la capacidad de transferencia en dispositivos. En el siguiente capítulo se dará a conocer los principios de la electrónica digital aplicada a la ingeniería de hardware.

EDITORIAL
MACRO®

Capítulo 3

Circuitos lógicos



EDITORIAL
MACRO®

Los circuitos lógicos son arreglos que tienen una secuencia de estados o niveles de tensión para dar solución a problemas de electrónica. La implementación de los circuitos lógicos se lleva a cabo mediante el uso de compuertas lógicas.

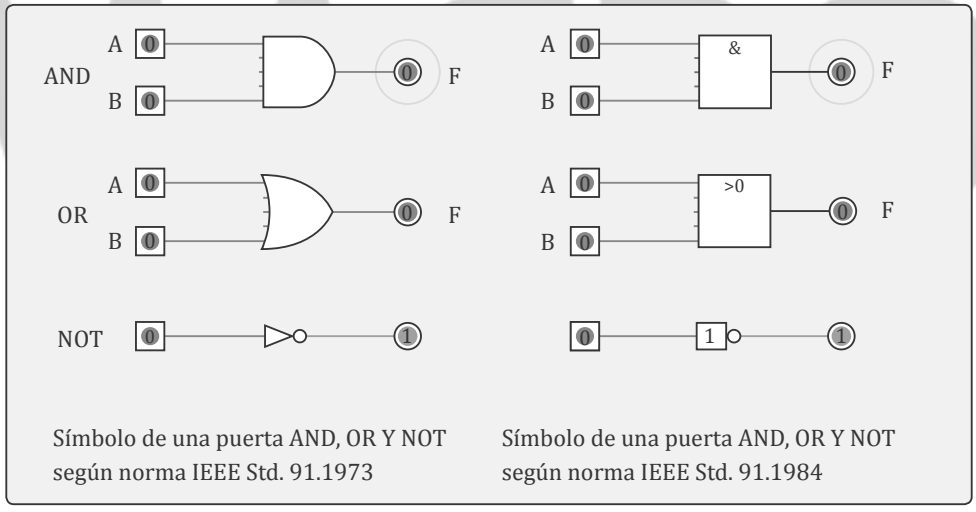
Actualmente, existen varias puertas lógicas, las cuales permiten comprender el funcionamiento de un ordenador convencional. Son las encargadas de llevar a cabo un conjunto de operaciones binarias y, además, de efectuar el proceso de la información mediante la construcción de circuitos digitales. En este capítulo se conocerá los tipos de señales, las puertas lógicas más utilizadas y su forma de aplicación en el ámbito de la ingeniería de hardware.

3.1 SEÑALES ANALÓGICAS Y DIGITALES

Como se ha mencionado una señal es una secuencia de valores que varía respecto al tiempo. Por lo general, existen dos tipos de señales: analógica y digital, ambas con ciertas características y diferencias. Las señales electrónicas se encuentran presentes en todo elemento que permite tanto la recepción como transmisión de datos. Las señales se encuentran hoy presentes en el ámbito de la ingeniería de hardware y el desarrollo, pues gracias a estas es posible la obtención de valores significativos fáciles de graficar e implementar.

3.2 COMPUERTAS LÓGICAS BÁSICAS

Las compuertas básicas para la construcción de circuitos lógicos son: AND, OR y NOT, estas se representan mediante símbolos. La simbología más utilizada al momento de representar un circuito lógico es la simbología estándar americana MIL (*military standard graphics symbols for logic diagrams*). Posteriormente, aparece una nueva simbología lógica, reconocida por las normas españolas, nombrada simbología IEC (*international electrotechnical commission*). Estas simbologías representan un estándar (IEEE) y ambas son empleadas en el ámbito de la electrónica digital.



Símbolo de una puerta AND, OR Y NOT según norma IEEE Std. 91.1973

Símbolo de una puerta AND, OR Y NOT según norma IEEE Std. 91.1984

Figura 3.1 Simbología de las tres compuertas lógicas básicas

Las compuertas lógicas tienen asociada una operación aritmética básica. La puerta AND tiene asociada la operación de multiplicación de bits; mientras que la puerta OR, la operación de suma. La inversión es característica de la compuerta NOT.

Las compuertas lógicas AND y OR, para este caso, pueden tener por lo menos dos entradas hasta N, aunque solo una salida X. En tanto, la compuerta NOT posee solamente una entrada y, por consiguiente, una salida.

Las entradas y salidas de una compuerta se representan mediante letras, las cuales juegan un papel muy importante para la construcción de arreglos que se conocen como tablas de verdad.

3.2.1 Tablas de verdad

Estas tablas se conforman por identificadores o entradas, además, de un conjunto de valores binarios debidamente ordenados. Las tablas de verdad se diseñan como esquema para la construcción de circuitos lógicos. Cada compuerta tiene su propia tabla de verdad:

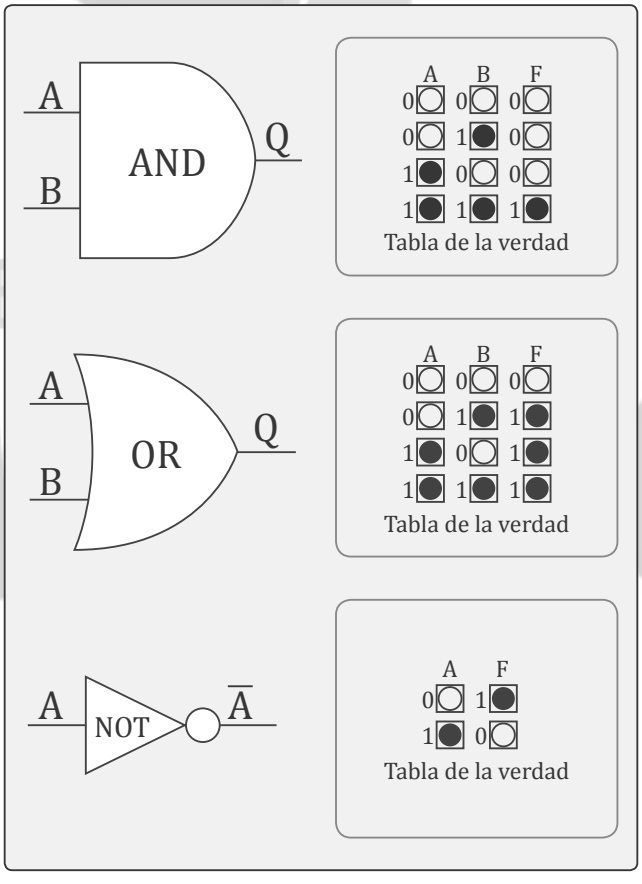


Figura 3.2 Tablas de verdad de cada compuerta lógica básica

Para la construcción de tablas de verdad de un circuito lógico funcional, se debe tener en cuenta la siguiente premisa:

Por un lado, el número de columnas en una tabla de verdad = las entradas existentes + 1 (la columna de la salida). El número de filas representa la cantidad de combinaciones en las entradas.

Por otro lado, el número de combinaciones = 2^N.

Donde N es el número de columnas de la tabla de verdad (menos la columna de salida).

	Entradas				Salidas	
Combinaciones	A	B	C	D	a	b
	0	0	0	0	1	1
	0	0	0	1	0	1
	0	0	1	0	1	1
	0	0	1	1	1	1
	0	1	0	0	0	1
	0	1	0	1	1	0
	0	1	1	0	1	0
	0	1	1	1	1	1
	1	0	0	0	1	1
	1	0	0	1	1	1
	1	0	1	0	X	X
	1	0	1	1	X	X
	1	1	0	0	X	X
	1	1	0	1	X	X
	1	1	1	0	X	X
	1	1	1	1	X	X

Figura 3.3 Ejemplo de tabla de verdad con cuatro entradas y dos salidas
Fuente: el autor.

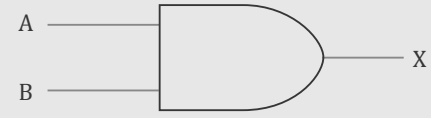
ACTIVIDAD 1

1. Cree una tabla de verdad dada la siguiente función: F = (a . b) (c . d).
2. Cree una tabla de verdad dada la siguiente función: F = (w + x) + (y . z).
3. ¿Cuántas combinaciones arroja la tabla de la siguiente función F = (x + y) (z . x . y)?
4. ¿Cuántas combinaciones arroja la tabla de la función F = (A . B) (C . D . E' . F')?

ACTIVIDAD 2

1. Dibuje la tabla de verdad para una puerta NOT.
2. Dada la siguiente figura, ¿cuál será el tren de salida?

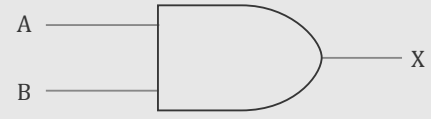
h)	g)	f)	e)	d)	c)	b)	a)
0	1	0	1	1	0	0	1



Pulso a:	Pulso c:	Pulso e:	Pulso g:
Pulso b:	Pulso d:	Pulso f:	Pulso h:

3. Respecto a la siguiente figura, ¿cuál será el tren de salida?

h)	g)	f)	e)	d)	c)	b)	a)
0	1	0	1	1	0	0	1
0	0	0	1	1	1	1	0



Pulso a:	Pulso c:	Pulso e:	Pulso g:
Pulso b:	Pulso d:	Pulso f:	Pulso h:

3.2.2 Construcción de circuitos lógicos

Para entender el contexto de la creación de circuitos lógicos, es necesario atender las siguientes premisas:

a. Los circuitos lógicos se crean a partir de una expresión booleana o en su defecto de una tabla de verdad.

b. Las expresiones booleanas pueden construirse a partir de un diagrama lógico, un diagrama topológico o una tabla de verdad.

Ejemplo de construcción de una expresión a partir de un circuito lógico

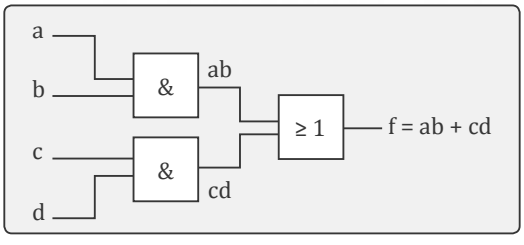


Figura 3.4 Representación de un circuito lógico y obtención de una expresión booleana

Ejemplo de construcción de una expresión a partir de una tabla de verdad

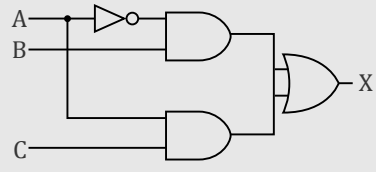
	A	B	C	F	
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	1	$\longrightarrow \overline{A}\ B\ \overline{C}$
3	0	1	1	1	$\longrightarrow \overline{A}\ B\ C$
4	1	0	0	1	$\longrightarrow A\ \overline{B}\ \overline{C}$
5	1	0	1	1	$\longrightarrow A\ \overline{B}\ C$
6	1	1	0	1	$\longrightarrow A\ B\ \overline{C}$
7	1	1	1	1	$\longrightarrow A\ B\ C$

$$F = \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$$

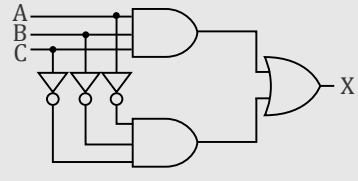
Figura 3.5 Construcción de una expresión lógica a partir de una tabla de verdad

ACTIVIDAD 3

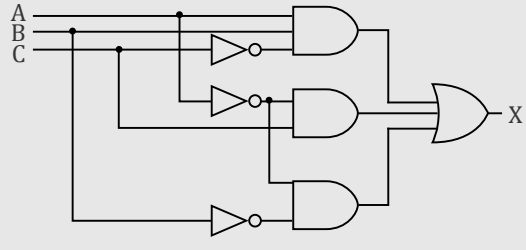
1. ¿Cuál es la tabla de verdad para la siguiente figura?



2. ¿Cuál es la expresión booleana y la tabla de verdad para el siguiente diagrama lógico?



3. ¿Cuál es la expresión booleana y la tabla de verdad para el diagrama lógico de la siguiente figura?



4. Trace el diagrama lógico y la tabla de verdad de las siguientes expresiones:
   - **a)** F = AB' + 'BC
   - **b)** F = A'BC' + 'BC'

3.3 COMBINACIÓN DE COMPUERTAS

Las compuertas básicas, según Thomas L. Floyd[1] se pueden combinar entre sí, dando lugar a circuitos electrónicos más complejos y, por lo tanto, a nuevas compuertas lógicas. Por ejemplo: una compuerta AND puede combinarse con una puerta NOT, lo que ofrece como resultado la compuerta NAND. Por otro lado, si se efectúa una combinación de una compuerta OR con NOT, se tendrá una compuerta NOR.

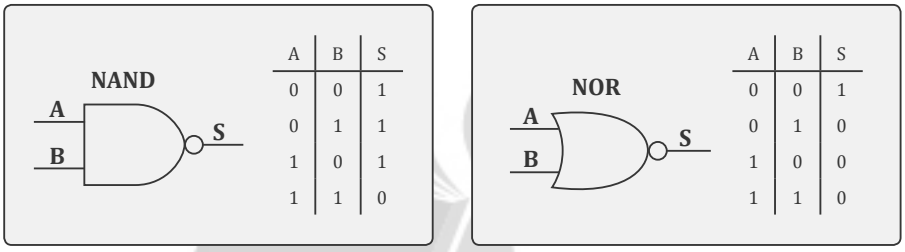
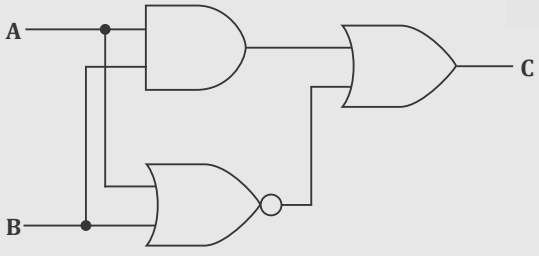


Figura 3.6 La posibilidad del uso de compuertas NAND y NOR con la combinación de compuertas lógicas básicas
Fuente: el autor.

ACTIVIDAD 4

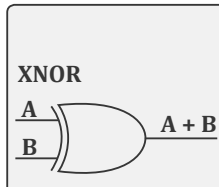
1. A partir del siguiente diagrama lógico, obtenga su expresión booleana equivalente:



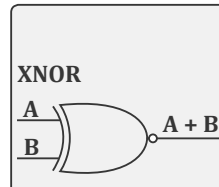
2. A partir de las siguientes expresiones lógicas, obtenga su diagrama lógico correspondiente:
   - **a)** F = AB + (A'B')' + (A + B)
   - **b)** F = (xy'z')' + (xy) + (x'z)' + x
   - **c)** F = AB'+ AC (BC)''

[1] Autor del libro *Fundamentos de sistemas digitales* (2006). Un material idóneo para quienes comienzan en el campo de estudio de la electrónica digital.

Otras compuertas muy utilizadas en el ámbito de la electrónica digital son la puerta XOR (o EXOR) y XNOR (o EXNOR).



Entrada A	Entrada B	Salida A + B
0	0	0
0	1	1
1	0	1
1	1	0



Entrada A	Entrada B	Salida A + B
0	0	1
0	1	0
1	0	0
1	1	1

Figura 3.7 Símbolo y tabla de verdad de la compuerta XOR y XNOR, respectivamente

Cada compuerta debe tener su propia expresión booleana. Por ejemplo, para la compuerta XOR Y XNOR respectivamente, se expresa lo siguiente:

XOR $S = A \oplus B =$ A B' + A' B

XNOR $S = \overline{A \oplus B} =$ A B + A' B'

3.4 SIMULACIÓN CON LOGISIM

Logisim es una herramienta de software que brinda la posibilidad de trazar diagramas lógicos y efectuar el análisis combinacional de cualquier circuito electrónico. Se trata de un simulador que no puede hacer falta a ningún estudiante del ámbito de la electrónica y la ingeniería de hardware. Permite desde la creación de tablas de verdad hasta el diseño de mapas de Karnaugh de manera automática, y la importación de extensiones y librerías que se integran fácilmente a cualquier proyecto.

Logisim es una herramienta ideal para la comprobación de resultados, montaje de expresiones, validación de diagramas y diseño de proyectos.

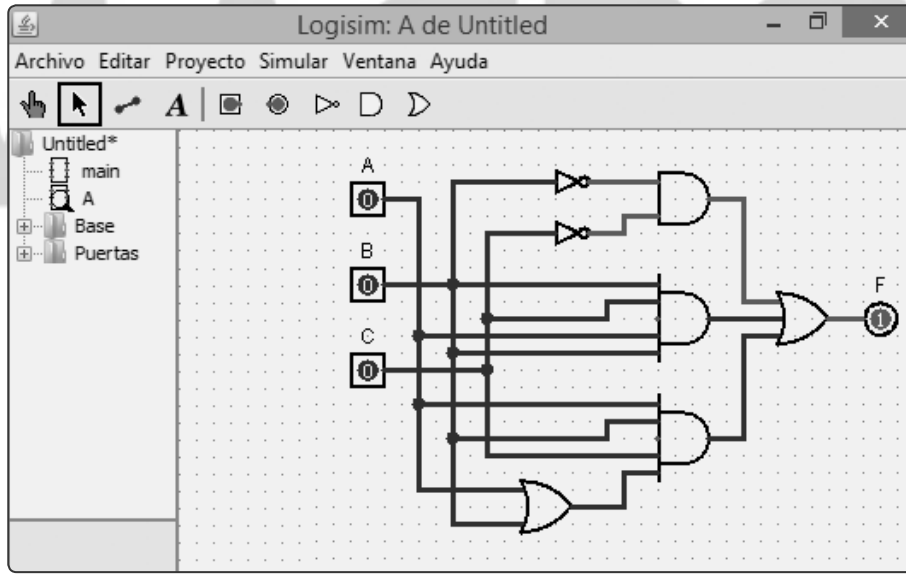


Figura 3.8 Diseño y simulación de una diversa gama de circuitos electrónicos a través de la herramienta logisim

3.5 EL CIRCUITO INTEGRADO

El circuito integrado es también llamado chip. Existen diferentes tipos de empaquetado y pueden efectuar distintas funciones. Actualmente, diminutos circuitos integrados (CI) funcionan como puertas o compuertas lógicas. Un tipo popular de transistores se denomina CI *dual-in-line package* (DIP) empaquetamiento de doble línea. Este CI particular se denomina DIP.

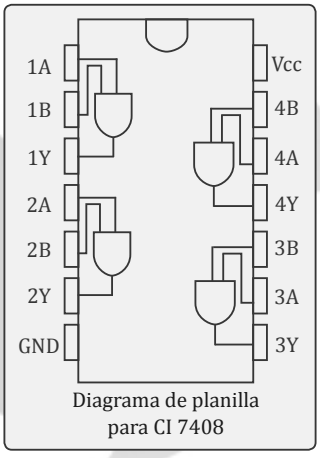


Figura 3.9 Esquema de conexión interna de un CI que implementa una compuerta AND de 2 entradas (x4)

Los circuitos integrados son componentes electrónicos, constituidos por compuertas lógicas. Hoy, los microprocesadores para PC contienen millones de CI trabajando en conjunto. En la figura anterior, se puede observar el esquema interno de un circuito integrado. En este caso, se encuentra conformado por cuatro compuertas AND de dos entradas y una salida cada una. Este arreglo recibe el nombre de CI-74LS08.

Los fabricantes de los CI proporcionan diagramas de plantillas similares a la figura anterior. Para conocer los datos proporcionados por estas plantillas, se aconseja la búsqueda de la hoja de datos o *datasheet* desde Internet.

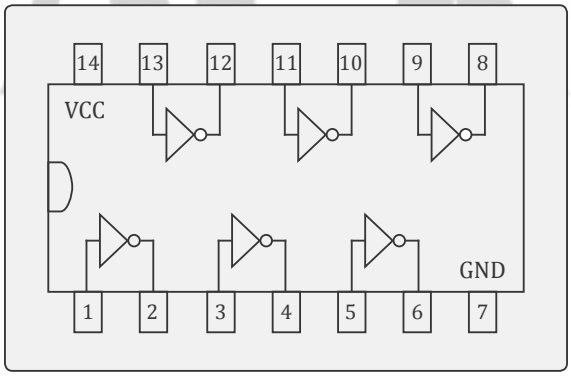


Figura 3.10 Esquema de conexión interna de un CI que implementa una compuerta NOT (x6)

Ejemplo de aplicación real de un CI (usando componentes electrónicos). Como se puede observar, las entradas (1 y 2, respectivamente) se encuentran representadas con *switches*, mientras que las salidas se ven como ledes.

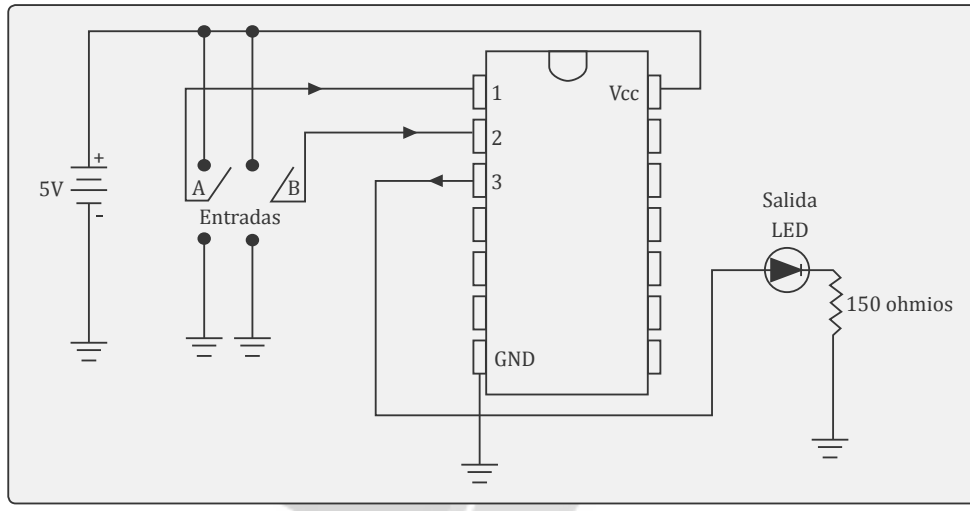


Figura 3.11 Diagrama de conexión de un circuito integrado (puerta lógica)

El diagrama anterior se conoce como diagrama topológico, sirve para representar un circuito electrónico terminado. Este tipo de diagramas se generan a partir de una expresión lógica, un diagrama lógico o una tabla de verdad.

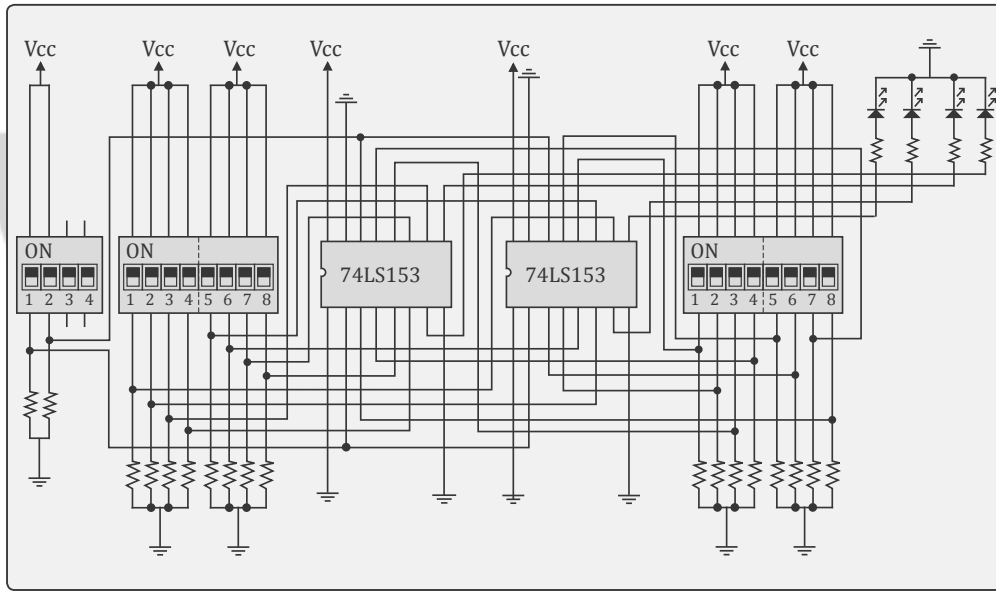


Figura 3.12 Ejemplo de un diagrama topológico que incluye tres DIP *switches* y dos CI

3.5.1 Simulación

Para el diseño de diagramas lógicos y topológicos se puede hacer uso de simuladores complejos. Con ello, se pretende poner en práctica cada uno de los ejercicios planteados antes de ser armados sobre una placa de prueba (comúnmente, llamada protoboard o breadboard). Después de este proceso, el prototipo puede ser grabado sobre una placa fenólica.

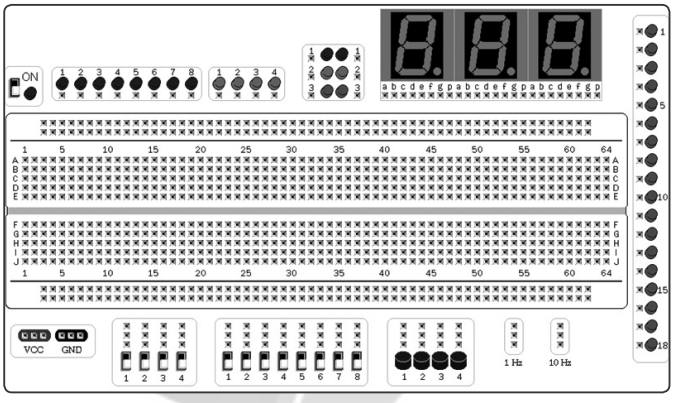


Figura 3.13 Interfaz de la herramienta constructora de circuitos (protoboard virtual)

Muchos ejercicios pueden ser probados desde este simulador de circuitos digitales. Su uso es muy sencillo, se limita a la conexión y armado de un circuito funcional. Está integrado por varios componentes electrónicos como ledes, pila (GND y VCC), interruptores o *switches*, displays y, desde luego, la placa de prueba. Otra solución muy parecida al constructor de circuitos es *Winbreadboard*.

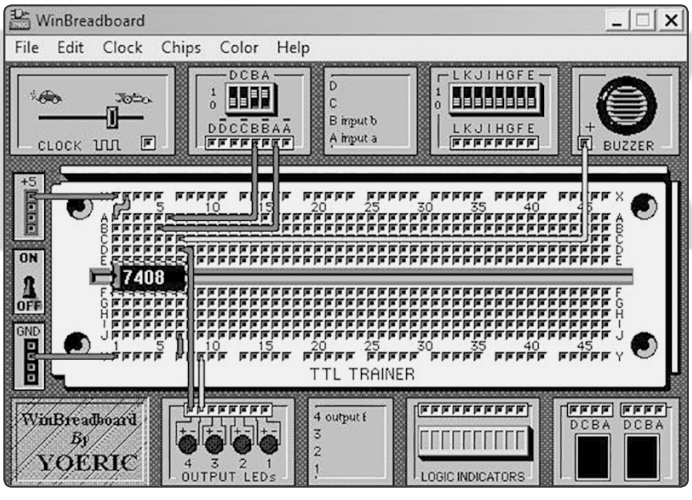


Figura 3.14 Ejemplo de uso del simulador o constructor de circuitos (protoboard virtual)

3.6 TECNOLOGÍA TTL

Con anterioridad, se ha hecho mención sobre la existencia de algunas tecnologías presentes en circuitos integrados. La tecnología o familia CMOS y la tecnología TTL.

La familia TTL implementa un conjunto de series (que habitualmente se presentan a modo de serigrafía impresa) que tienen como objetivo identificar ciertos componentes electrónicos como un CI. Frecuentemente, resulta ideal conocer y familiarizarse con estas series, pues, en más de una ocasión será necesario recurrir a dichos datos para solicitar el material de electrónica que ha pedido el profesor o, simplemente, para reconocer la lista de materiales que necesitará para el montaje de cualquier proyecto. Para comprender mejor lo anterior, se propone el análisis de la siguiente figura y un ejemplo:

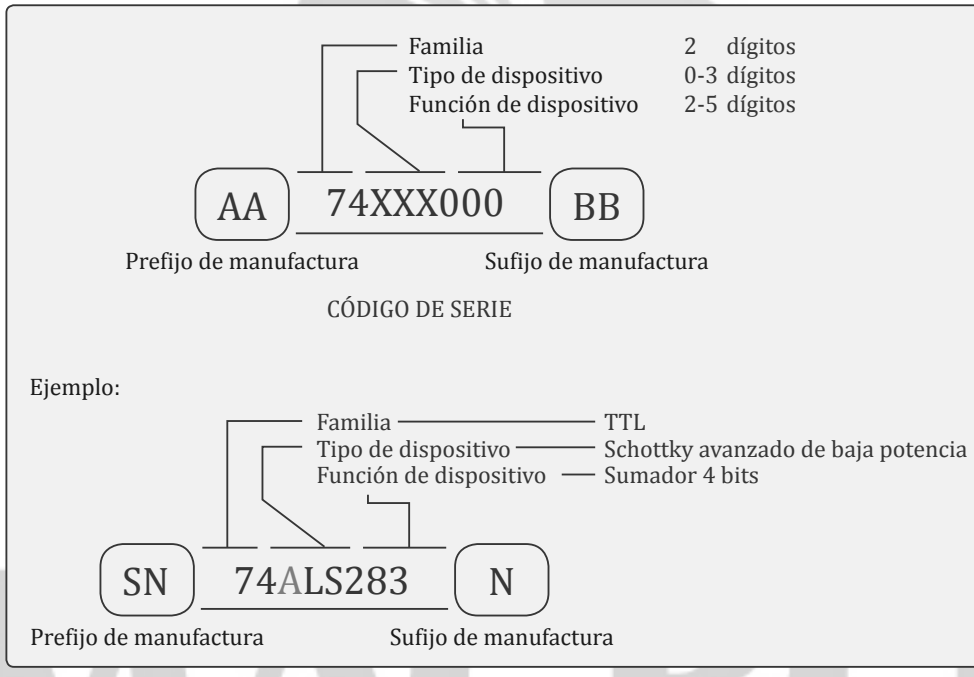


Figura 3.15 Esquema de rotulación de un CI 74283 (sumador binario)

Para su comprensión se parte del siguiente ejemplo: se tiene un circuito integrado cuya inscripción es DIP-CI SN74ALS283N, el término DIP, hace referencia a la sigla *Dual In-line Package* (empaquetado de entrada en línea doble). Se trata de un tipo de empaquetado especial de fabricación, el cual, generalmente, consta de un conjunto de contactos o pines de conexión en arreglo dual.CI, significa circuito integrado, mientras que 74ALS (TTL Schottky avanzado de baja potencia) hace referencia a un circuito de la familia TTL de conmutación de baja potencia. Los últimos dígitos representan el número de identificación del circuito, en este caso un sumador binario de 4 bits. El número que identifica la función del CI puede llegar a variar, según la necesidad del proyecto.

En la siguiente tabla se muestra la información necesaria sobre consumo de la familia TTL:

Tabla 3.1 *Características y atributos de la familia TTL*

Parámetro	Unidad	74	74L	74H	74S	74LS	74AS	74ALS
t_p	ns	9	33	6	3	9	1.6	5
P_d	mW	10	1	22	20	2	20	1.3
P. V	pJ	90	33	132	60	18	32	6.5
V_{ILmax}	V	0.8	0.7	0.8	0.8	0.8	0.8	0.8
V_{OLmax}	V	0.4	0.4	0.4	0.5	0.5	0.5	0.5
V_{IHmin}	V	2.0	2.0	2.0	2.0	2.0	2.0	2.0
V_{OHmin}	V	2.4	2.4	2.4	2.7	2.7	2.7	2.7
I_{ILmax}	mA	-1.6	-0.18	-2.0	-2.0	-0.4	-2.0	-0.2
I_{OLmax}	mA	16	3.6	20	20	8	20	8
I_{IHmax}	µA	40	10	50	50	20	200	20
I_{OHmax}	µA	-400	-200	-500	-1000	-400	-2000	-400

P_d = Consumo de potencia por compuerta, P. V = producto de potencia-velocidad

ACTIVIDAD 5

1. Dado el número de serie de un CI, determine el nombre válido de la compuerta que lo implementa. Escriba también el número de entradas por compuerta lógica.
   - **a)** CI-74LS08: _____
   - **b)** CI-74LS21: _____
   - **c)** CI-74LS04: _____
   - **d)** CI-74LS00: _____
   - **e)** CI-74LS32: _____
   - **f)** CI-74LS386: _____
   - **g)** CI-74LS86: _____
   - **h)** CI-74LS30: _____

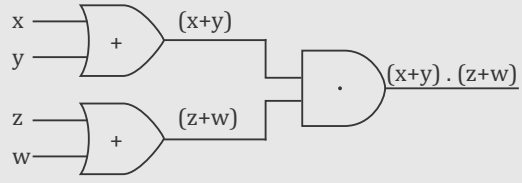
2. Dado el nombre y número compuertas lógicas, colocar el nombre válido del CI.
   - **a)** AND (x3): _____
   - **b)** NOR (x3): _____
   - **c)** NAND (x3): _____
   - **d)** NOT (x6): _____
   - **e)** NAND (x2): _____
   - **f)** NOR (x4): _____

3. Realice el esquema interno de los siguientes circuitos y su tabla de verdad:
   CI-7408, CI-7432, CI-7404, CI-7486, CI-74386, CI-7400, CI-7402.

ACTIVIDAD 6

Diagrama 1

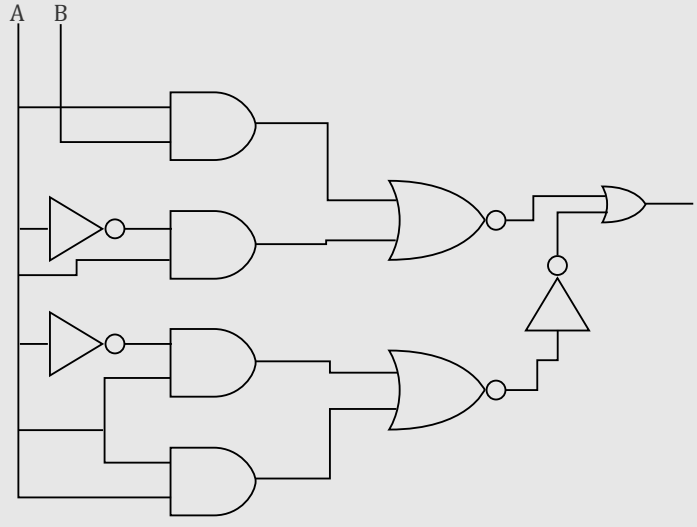
1. Mencione el número de serie de los CI que utilizaría para armar el siguiente circuito:
2. ¿Cuántos *switches* se necesitan?
3. ¿Cuántos ledes se necesitan?



4. Genere la expresión lógica, el diagrama topológico y la tabla de verdad del presente diagrama. Compruébelo bajo simulación.

Diagrama 2

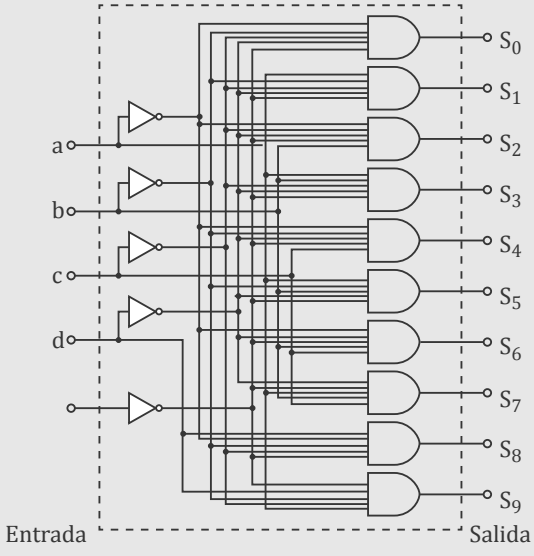
1. Mencione el número de serie de los CI que utilizaría para armar el siguiente circuito:
2. ¿Cuántos *switches* se necesitan?
3. ¿Cuántos ledes se necesitan?



4. Genere la expresión lógica, el diagrama topológico y la tabla de verdad del presente diagrama. Compruébelo bajo simulación.

Diagrama 3

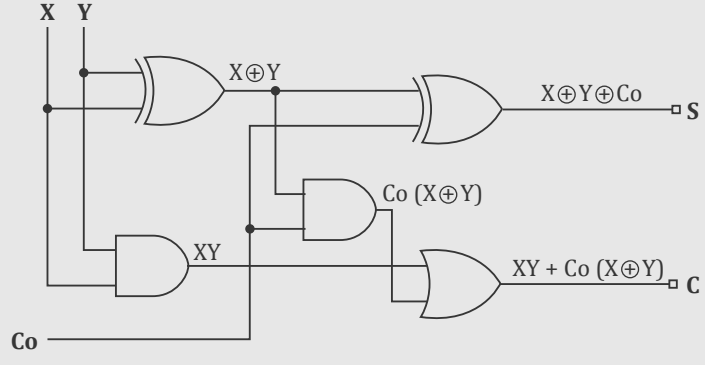
1. Mencione el número de serie de los CI que utilizaría para armar el siguiente circuito:
2. ¿Cuántos *switches* se necesitan?
3. ¿Cuántos ledes se necesitan?



4. Genere la expresión lógica, el diagrama topológico y la tabla de verdad del presente diagrama. Compruébelo bajo simulación.

Diagrama 4

1. Mencione el número de serie de los CI que utilizaría para armar el siguiente circuito:
2. ¿Cuántos *switches* se necesitan?
3. ¿Cuántos ledes se necesitan?



4. Genere la expresión lógica, el diagrama topológico y la tabla de verdad del presente diagrama. Compruébelo bajo simulación.

Práctica de laboratorio n.º 2

Manejo de compuertas lógicas

Objetivo de la práctica: Comprobar las tablas funcionales o de verdad de los componentes básicos Y (AND), O (OR), NO (NOT), NO-Y (NAND), NO-O (NOR), O-EXCLUSIVA (OREX) y NO-O-EXCLUSIVA (NOREX), utilizando circuitos integrados.

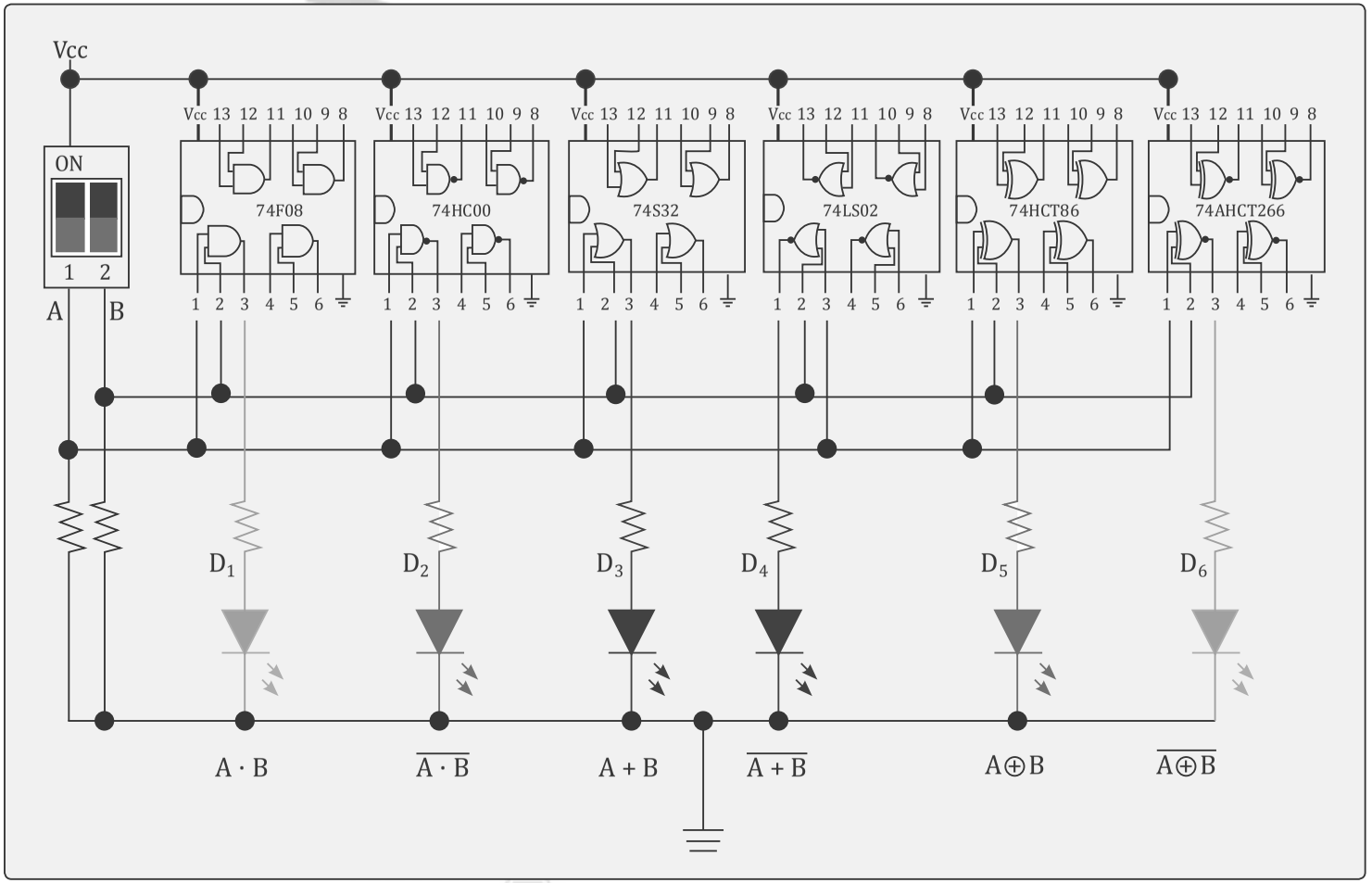
Duración: 1 hora.

Materiales:

- Fuente de voltaje de 5 V.
- Un DIP de 8 entradas.
- 6 ledes (diodo emisor de luz, por sus siglas en inglés), no importa el color.
- 8 resistencias de 470 Ohmios.
- Una tablilla de conexiones (protoboard).
- Los siguientes circuitos integrados o equivalentes:
  - 74F08 (4 compuertas Y de 2 entradas).
  - 74H00 (4 compuertas NO-Y de 2 entradas).
  - 74H04 (6 compuertas NO).
  - 74S32 (4 compuertas O de 2 entradas).
  - 74LS02 (4 compuertas NO-O de 2 entradas).
  - 74HCT86 (4 compuertas O EXC de 2 entradas).
  - 74AHCT266 (4 compuertas NO-O-EXC de 2 entradas).
- Alambre para conexiones (JUMPERS).

Procedimiento experimental

Armar el siguiente circuito topológico para comprobar las tablas de verdad.



Cuestionario inicial

Conteste las siguientes preguntas con respecto a lo visto en el montaje anterior.

1. ¿Qué es VCC y GND?

   _____

2. ¿Cómo distingue el polo negativo del positivo en un led?

   _____

   _____

   _____

3. ¿Cuál es el pin reservado para GND y para VCC en un CI y cómo se conectan sobre protoboard?

   _____

   _____

   _____

4. ¿Para qué sirve un protoboard?

   _____

   _____

   _____

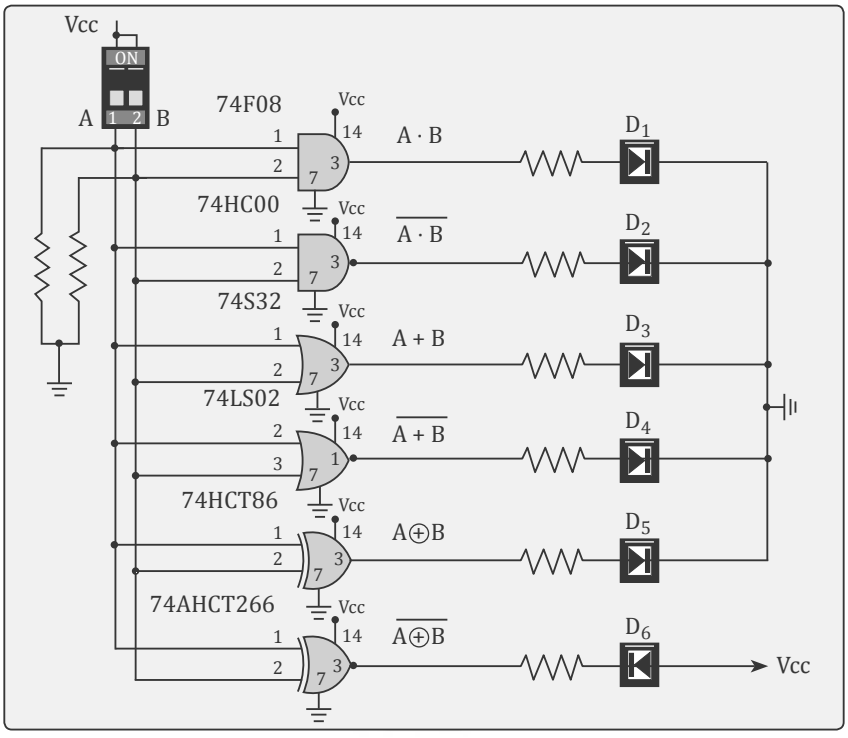
5. ¿Qué aprendizaje obtuvo del anterior montaje?

   _____

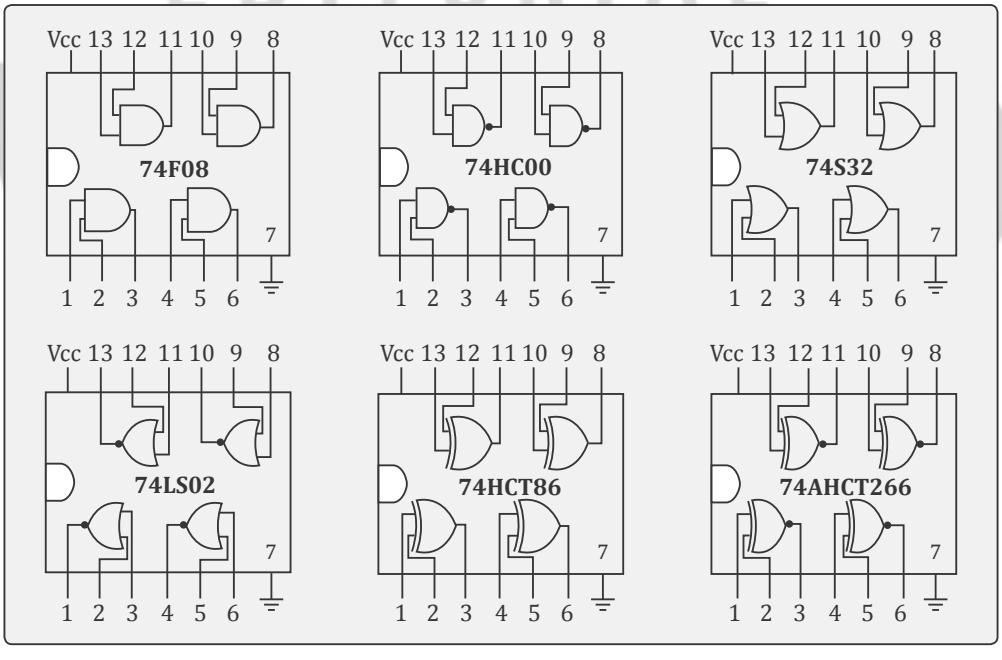
   _____

   _____

El circuito topológico también puede presentarse de la siguiente forma equivalente:



Se observa que en el led D^1 se comprobará la compuerta Y de dos entradas; en D^2, la tabla de verdad de la compuerta NO-Y de 2 entradas y así, sucesivamente. A continuación, se muestra la configuración interna de los CI usados en los diagramas anteriores.



Cuestionario final

1. En una compuerta Y de 2 entradas; si en una de sus entradas recibe un 0 y en la otra un 1, ¿cuál es su salida?

   _____

2. Si una compuerta NO-Y recibe las mismas señales de entrada de la pregunta anterior, ¿cuál es su salida?

   _____

3. Si a una compuerta O llegan a sus entradas 2 unos, ¿cuál es su salida?

   _____

4. Si en el circuito de la práctica se desconectan las entradas 1 y 2 del DIP, ¿qué es lo que pasa en los diodos emisores de luz (led)? ¿Por qué?

   _____

   _____

5. En un circuito integrado TTL (*transistor transistor logic*, lógica-transistor-transistor), en las entradas de cualquier compuerta, por definición, se considera ¿un 1 o un 0?

   _____

6. ¿Qué es lo que pasa con un led si se conecta en polarización inversa?

   _____

   _____

7. ¿A qué rango de voltaje se le considera un 1 lógico?

   _____

8. ¿A qué rango de voltaje se le considera un 0 lógico?

   _____

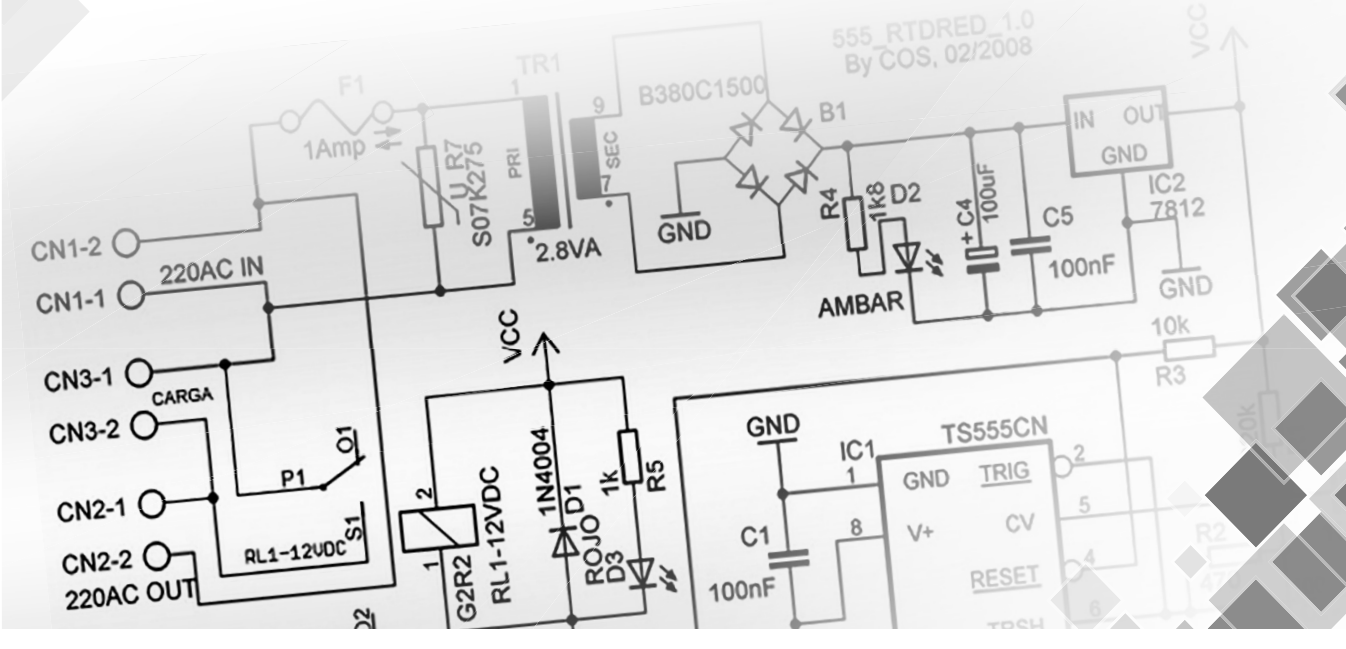
9. Realizar el montaje de las compuertas lógicas antes empleadas sobre *LiveWire* para comprobar su funcionamiento.

En este capítulo se ha revisado a detalle todo lo relacionado con la creación de circuitos digitales (implementación de compuertas lógicas), desde el diseño de logigramas, formulación de expresiones booleanas, desarrollo de tablas de verdad y construcción de diagramas topológicos. Se ha desarrollado también la forma de conexión de un circuito sobre protoboard mediante el uso de circuitos integrados. En el siguiente capítulo se darán a conocer los métodos para la reducción de circuitos lógicos.

EDITORIAL
MACRO®

Capítulo 4

Reducción de circuitos electrónicos



EDITORIAL
MACRO®

La reducción de circuitos consiste en una técnica utilizada en la electrónica para simplificar tanto el desarrollo como la implementación de un circuito electrónico funcional. Para llevar a cabo esta tarea debe emplearse algún método adecuado de reducción.

Para la reducción de circuitos, por lo general, deberá partirse de una expresión lógica válida (aunque, no debería suponer ningún problema si se tiene a la mano algún diagrama o tabla de verdad, que servirán de base para la generación de expresiones booleanas), la cual deberá adoptar un método para la simplificación. De este modo, se obtiene una función resultante (más pequeña) derivada de la expresión original.

El propósito principal del empleo de técnicas para la reducción de circuitos se centra en la optimización de costos tanto económicos como los implicados en tiempos de implementación.

4.1 MÉTODOS DE REDUCCIÓN

Existen dos métodos para la reducción de funciones o expresiones booleanas y, por consiguiente, de diagramas lógicos. Se trata del álgebra de Boole y los mapas de Karnaugh.

El álgebra de Boole establece varios teoremas, leyes y postulados que hacen posible la reducción de un circuito electrónico. Mientras que, los mapas de Karnaugh representan una técnica que hace uso de un tipo especial de tabla, la cual se encuentra, estratégicamente, dividida en posiciones para la reducción. En este capítulo, se propone el desarrollo de ambos métodos de simplificación.

4.1.1 El álgebra de Boole

El álgebra de Boole consiste en un conjunto de elementos que pueden tomar dos valores 0 y 1, los cuales están asociados a dos operaciones binarias denominadas suma y producto lógico.

A. Propiedades del álgebra de Boole

El álgebra de Boole posee varias propiedades, entre las que se encuentran:

a. Propiedad conmutativa: Para entender el contexto de esta propiedad se tienen dos premisas.

1. Para la suma lógica, el orden de los sumandos no altera la suma.
2. Para el producto lógico, el orden de los factores no altera el producto.

Si a y b son elementos del álgebra, se verifica:

1	2
$a + b = b + a$	$a\,b = b\,a$

b. Propiedad asociativa (precedencia):

1. Al aplicar la operación OR a más de dos variables, el resultado es el mismo, independientemente de la forma en que se agrupen las variables. Es parecida a la ley conmutativa.
2. Al aplicar la operación AND a más de dos variables, el resultado es el mismo, independientemente de la forma en que se agrupen las variables.

1	2
(a + b) c = a + (b + c) = a + b + c	(a b) c = a (b c) = a b c

c. Propiedad de identidad o elemento neutro: Dentro del álgebra de Boole, tanto el 0 como el 1 lógico son tratados como un elemento neutro. Cumplen la propiedad de identidad con respecto a cada una de dichas operaciones:

0 + a = a	1 a = a

d. Propiedad distributiva: Consiste en obtener el factor común, en el que la variable que más se repite en la expresión se extrae como factor de los productos parciales.

a (b + c) = a b + a c	a + (b . c) = (a + b) (a + c)

e. Propiedad de inversión o complemento: Consiste en efectuar una inversión o complemento de una variable. Para cada elemento a del álgebra existe un elemento denominado a, tal que:

a + a' = 1	a a' = 0

f. Principio de la dualidad: Este principio establece que cualquier teorema o identidad algebraica deducible de los postulados anteriores puede transformarse en un segundo teorema o identidad válida sin más que intercambiar (+) por (*) y 1 por 0.

B. Reglas básicas del álgebra de Boole

Resultan muy útiles para la reducción de expresiones. Las literales empleadas pueden representar una única variable o una combinación de variables.

La operación AND o Y	
0 ■ 0 = 0	0 ■ 0 = 0
0 ■ 1 = 0	0 ■ A = A
1 ■ 0 = 0	A ■ 0 = 0
1 ■ 1 = 1	A ■ A = A

La operación OR o O	
0 + 0 = 0	A + 0 = A
0 + 1 = 1	A + 1 = 1
1 + 0 = 1	A + A = A
1 + 1 = 1	A + A = 1
La operación NOT o No	
$\overline{0} = 1$	A'' = A
$\overline{1} = 0$	Nota: A' = $\overline{A}$

Por lo general, los circuitos de conmutación o lógicos son representados en diagrama de contacto o diagrama de llaves (circuitos en serie o en paralelo). Esta representación es frecuentemente empleada para memorizar, más fácilmente, los postulados o leyes del álgebra de Boole antes descritas y los teoremas.

Se sabe que la operación de suma se asimila a la conexión en paralelo, mientras que la operación de producto lógico se asemeja a la conexión en serie.

El elemento 0 es un contacto (o interruptor) que está siempre abierto, en tanto que el elemento 1 representa un contacto que está siempre cerrado.

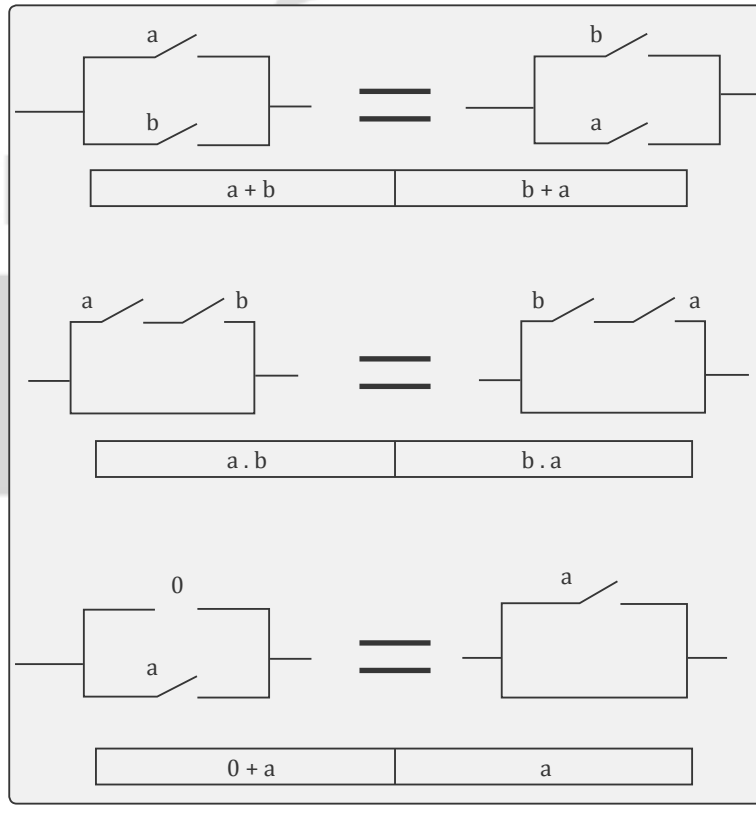


Figura 4.1 Representación de una expresión booleana en diagrama de contactos

C. Teoremas de álgebra de Boole

Un teorema es definido como una proposición matemática demostrable a partir de axiomas o de proposiciones ya demostradas. El álgebra de Boole propone los siguientes teoremas:

a. Teorema 1: El elemento complemento A' es único.

b. Teorema 2: Para cada elemento a del álgebra de Boole se verifica:

a + 1 = 1	a . 0 = 0

c. Teorema 3 (idempotencia): La idempotencia es la propiedad para realizar **una acción un determinado número de veces**, y asegurar un mismo resultado por cada intento. Para cada elemento a del álgebra de Boole se verifica:

a + a = a	a . a = a

d. Teorema 4 (involución): La involución es la propiedad que establece que la doble negación de una variable a, ofrece como resultado el valor original de a. Para cada elemento a del álgebra de Boole se verifica:

(a)´´ = a

e. Teorema 5 (absorción): Conocido también como ley de cancelación, la cual establece que por cada proceso de suma o producto se cancela el término independiente. Para cada par de elementos del álgebra de Boole a y b se verifica:

a + (ab) = a	a (a + b) = a

f. Teorema 6 (asociatividad): Cada uno de los operadores binarios (+) y (*) cumple la propiedad asociativa:

a + (b + c) = (a + b) + c = a + b + c	a (b c) = (a b) c = a b c

- **Leyes de De Morgan**

Esta ley se encarga de definir dos nuevas funciones lógicas de gran importancia, que serán utilizadas como elementos básicos para la realización de los sistemas digitales. Estas dos funciones se denominan NOR y NAND.

Las leyes de De Morgan declaran que la suma de n variables globalmente negadas es igual al producto de las N variables negadas individualmente. Asimismo, que el producto de n variables globalmente negadas es igual a la suma de las N variables negadas individualmente[1].

NAND	NOR
(a b)' = a' + b'	(a + b)' = a' b'

[1] Para mayor información sobre el tema, véase Thomas, F (2006), «Fundamentos de sistemas digitales».

Las funciones NOR y NAND de una sola variable constituyen la función de inversión. La realización de las funciones suma, producto e inversión con las funciones NOR y NAND, se representan mediante los símbolos antes analizados:

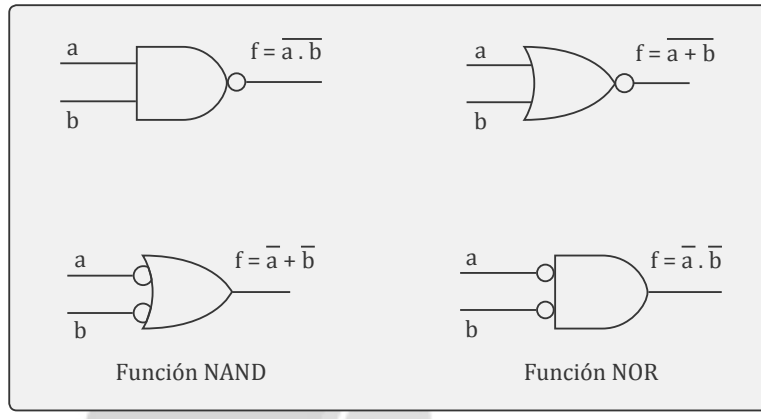


Figura 4.2 Simbología que representa las leyes de De Morgan

Los postulados anteriores tienen como único fin auxiliarnos para la simplificación. Para efectuar cualquier operación, es necesario contar con conocimientos sobre sistemas numéricos y manejo de operadores lógicos.

Para entender el presente contexto, se propone el desarrollo de algunos ejercicios de reducción de expresiones. Estos se muestran a continuación

Ejemplo de reducción de circuitos mediante el uso del álgebra de Boole

Ecuación original

$$F = (a\ b + a\ c)' + a'\ b'\ c$$

Reducción

$$F = a' + b'\ c'$$

Desarrollo aplicando las leyes y teoremas del álgebra de Boole

a. Ley de De Morgan

Antes	**(a b + a c)'** + a' b' c
Después	**(a' + b') (a' + c')**
En general	**(a' + b') (a' + c')** + a' b' c

b. Propiedad distributiva

Antes	**(a' + b') (a' + c')** + a' b' c
Después	**a' (a' + c') + b'(a' + c')**
En general	**a' (a' + c') + b'(a' + c')** + a' b' c

c. Propiedad distributiva

Antes	**a' (a' + c')** + b'(a' + c') + a' b' c
Después	**(a' a') + (a' c')**
En general	**(a' a') + (a' c')** + b'(a' + c') + a' b' c

d. Teorema idempotencia

Antes	**(a' a')** + (a' c') + b'(a' + c') + a' b' c
Después	**a'**
En general	**a'** + (a' c') + b'(a' + c') + a' b' c

e. Teorema absorción

Antes	**a' + (a' c')** + b'(a' + c') + a' b' c
Después	**a'**
En general	**a'** + b'(a' + c') + a' b' c

f. Propiedad distributiva

Antes	a'+ **b'(a' + c')** + a' b' c
Después	**b' a' + b' c'**
En general	a' + **b' a' + b' c'** + a' b' c

g. Propiedad distributiva

Antes	a' + b' a' + b' c' + a' b' c
Después	**b' a' + b' c'**
En general	a' + b' a' + b' c' + a' b' c

h. Teorema absorción

Antes	**a' + b' a' + b' c'** + a' b' c
Después	**a' + b' c'**
En general	a' + b' c' + a' b' c

i. Teorema absorción

Antes	**a' + b' c' + a' b' c**
Después	**a' + b' c'**
Resultado final	**F= a' + b' c'**

Nota de interés

App sobre álgebra de Boole

Entre los diferentes recursos *online* que existen se pueden ubicar algunas app (generalmente desarrolladas para Android) que permiten la reducción de expresiones booleanas. Una de ellas es Morgana, esta consiste en una aplicación muy sencilla de comprender que muestra la simplificación de manera muy intuitiva (resalta de colores las diferentes leyes, teoremas o postulados del álgebra de Boole). Ideal para cualquier estudiante de ingeniería.

ACTIVIDAD 1

1. Dadas las siguientes funciones lógicas, reduzca a su mínima expresión mediante álgebra de Boole.
   - **a)** F = (a b + a c c' + a' b + a b c b' + a b') (a c' + a' c' + c)
   - **b)** F = (x y +z) (y + y + z') + x y'
   - **c)** F = (a + b + c) (a b')'

2. Utilice la app Morgana para realizar los siguientes ejercicios, anote su resultado.
   - **a)** F= (x y + z y) + (x' y + z') + (x y')'
   - **b)** F = a + (b c +d) + a' + (b' c' d')
   - **c)** F= a c + a b) (a + b + c) (a b')

D. Representación de funciones algebraicas boolenas

Una función de álgebra de Boole es una variable binaria, cuyo valor es igual al de una expresión algebraica, en la que se relacionan entre sí las variables binarias por medio de las operaciones básicas (producto lógico, suma lógica e Inversión).

Se llama **término canónico** de una función lógica a todo producto o suma, en la cual aparecen todas las variables en su forma directa o inversa. Al primero de ellos se le llama producto canónico (minitérminos); al segundo, suma canónica (maxitérminos).

El **minitérmino** es un término producto, representado por un 1 lógico sobre una tabla de verdad. Este es identificado como una suma de productos (Sum Of Products o SOP).

Posición	a	b	C	F (a, b, c)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

F (a, b, c) = m0 + m2 + m3 + m7 = ∑m (0, 2, 3, 7)

F (a, b, c) = (a' b' c') + (a' b c') + (a' b c) + (a b c)

Las expresiones anteriores forman parte de una nomenclatura especial, creada para identificar la agrupación de variables en minitérminos o maxitérminos. A partir de este momento se conocerá como **nomenclatura de términos**.

Como se puede apreciar un minitérmino se identifica con el signo ∑ (sumatoria) y el parámetro m. Todos los números dentro del paréntesis son valores decimales que indican la posición en una tabla de verdad.

El **maxitérmino** es un término suma, representado por un 0 lógico sobre una tabla de verdad. Este es identificado como un producto de sumas (Product Of Sum o POS).

Posición	a	b	C	F (a, b, c)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

La nomenclatura utilizada para identificar un maxitérmino es la siguiente:

F (a, b, c) = M1 + M4 + M5 + M6 = ∏M (1, 4, 5, 6)

F (a, b, c) = (a + b + c') (a' + b + c) + (a' + b + c') + (a' + b' + c)

Como se puede apreciar un maxitérmino se identifica con el signo $\prod$ y el parámetro **M**.

En resumen, existen dos formas de poder reducir una expresión booleana, a través de minitérminos y maxitérminos.

Minterms = SOP (suma de productos) = Reducción por 1= $\sum$m (0, 2, 3, 7)

Maxterms = POS (producto de sumas) = Reducción por 0 = $\prod$M (1, 4, 5, 6)

Para mayor facilidad de representación, cada término canónico se expresa mediante un número decimal equivalente al binario obtenido al sustituir las variables ordenadas con un criterio determinado por un 1 o un 0, según aparezcan en su suma directa o complementaria respectivamente.

Por ejemplo, los términos canónicos siguientes representarán:

b' c d e' = 0110b = 6d

b + c'+ d + e' = 1010b = 10d

A continuación, se muestra la manera en que se puede obtener tanto una tabla de verdad como una expresión booleana en su forma canónica partiendo de la nomenclatura de términos.

a. Suma lógica: Dada la siguiente nomenclatura: F (a, b, c) = $\sum$ m (2,3,5), obtener tabla de verdad y expresión booleana en su forma canónica.

1. La tabla de verdad debe trazarse de acuerdo al número de entradas (columnas) haciendo uso la fórmula de 2 a la N (F (a, b, c)). En este caso, al realizar una reducción por minitérminos, deberán considerarse todas las salidas cuyo valor sea equivalente a un 1 lógico. Marcando el resto con una X.

Posición	a	b	c	F
0	0	0	0	X
1	0	0	1	X
2	0	1	0	1
3	0	1	1	1
4	1	0	0	X
5	1	0	1	1
6	1	1	0	X
7	1	1	1	X

2. La expresión booleana requiere de la formulación de términos en su forma canónica. Para ello, se utilizarán las posiciones cuya salida sea 1, las cuales se señalan en la siguiente expresión: ∑ m (2,3,5).

Posición	a	b	c	F	F
0	0	0	0	X	
1	0	0	1	X	
2	0	1	0	1	a' b c'
3	0	1	1	1	a' b c
4	1	0	0	X	
5	1	0	1	1	a b' c
6	1	1	0	X	
7	1	1	1	X	

3. Al realizar una expresión lógica por minitérminos (suma de productos), se obtiene:

   F = (a' b c') + (a' b c) + (a b' c) en su forma canónica.

b. Producto lógico: Dada la siguiente nomenclatura: F (a, b, c) = ∏ M (1,2,7), obtener tabla de verdad y expresión booleana en su forma canónica.

1. La tabla de verdad se obtiene de la misma manera que la suma lógica.
2. Para la formulación de la expresión booleana deben utilizarse las posiciones cuya salida sea 0. Las cuales, se señalan en la expresión: ∏ M (1,2,7). Una vez hecho esto, deberán invertirse las literales de cada término (los 0 por 1 y viceversa).

Posición	a	b	c	F	F
0	0	0	0	X	
1	0	0	1	0	a + b + c'
2	0	1	0	0	a + b' + c
3	0	1	1	X	
4	1	0	0	X	
5	1	0	1	X	
6	1	1	0	X	
7	1	1	1	0	a' + b' + c'

3. De esta manera, al formular una expresión lógica por maxitérminos (producto de sumas), se obtiene:

   F = (a + b +c') (a+b' +c) (a' +b' +c') en su forma canónica.

ACTIVIDAD 2

1. Dada las siguientes expresiones, simplifique por álgebra de Boole. Además, cree su diagrama lógico y su respectivo diagrama topológico tanto de la forma original como simplificada (puede auxiliarse de logisim).

   a) F = (A + B + C) (A + C)

   b) F = (A + BC')' (A B'+ BC)'

   c) F = A(AB)' + B(AB)'

2. Dada la siguiente nomenclatura de términos, obtenga la tabla de verdad, la expresión booleana correspondiente en su forma canónica y la expresión mínima.

   a) F (a, b, c) = ∏ (1, 3, 5)

   b) F (x, y, z) = ∑ (3, 4, 5)

   c) F (w, x, y, z) = ∏ (2, 4, 8, 14)

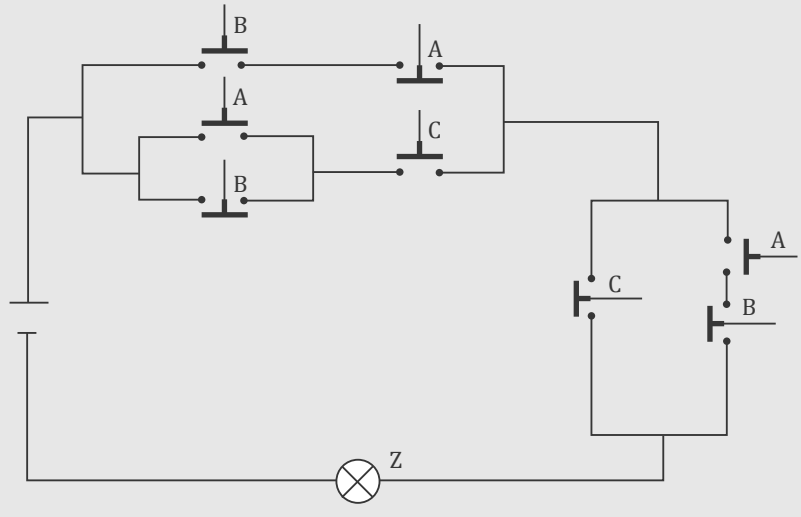
   d) F (A, B, C) = ∏ (0, 5, 6)

   e) F (a, b, c, d) =∑ (0, 2, 9, 10)

3. Dado el siguiente circuito, obtenga:

   a) Tabla de verdad, expresión booleana reducida y diagrama lógico correspondiente.

   b) Comprobar el ejercicio con la ayuda de la app Morgana.



Práctica de laboratorio n.º 3

Álgebra de Boole

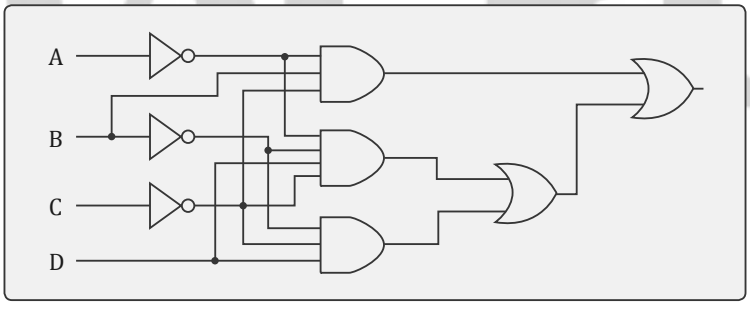
Objetivo de la práctica: Comprobar, en el laboratorio, el diseño optimizado de un circuito utilizando el álgebra de Boole y reportar las ventajas que se obtienen.

Duración: 1 hora.

Materiales:

- Una fuente de voltaje de 5V.
- 2 DIP 3 led (no importa el color).
- 11 resistencias de 470 ohm.
- 2 tablillas de conexiones.
- Los siguientes circuitos integrados:
  - Dos 74LS10 (3 compuertas NO-Y de 3 entradas.
  - Dos 74LS11.
  - Dos 74LS04.
  - Dos 74LS32 (4 compuertas O de 2 entradas)
  - Un 74LS21.
- Alambre para conexiones.

Dado el siguiente circuito lógico:



Cuya función de salida Z del circuito anterior es:

$$Z(A,B,C,D) = A'BC' + A'B'C'D + B'C'D$$

La tabla de verdad es:

A	B	C	D	A'BC'	A'B'C'D	B'C'D	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

Y el **circuito topológico** para generar **Z** es:

(Dibujar diagrama topológico, mismo que será armado sobre protoboard virtual y físico)

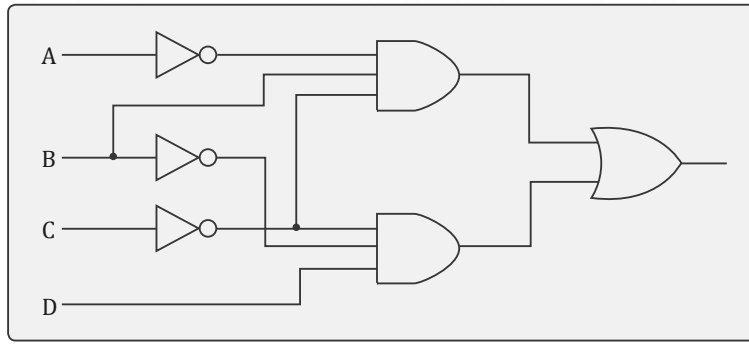
Simplificando Z, con el álgebra de Boole, se tiene:

Z (A, B, C, D) = A'BC' + A'B'C'D + B'C'D =

_____

_____

El logigrama de la función reducida del circuito es:



La tabla de verdad de la función reducida es:

A	B	C	D	A'BC'	B'C'D	Z
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0

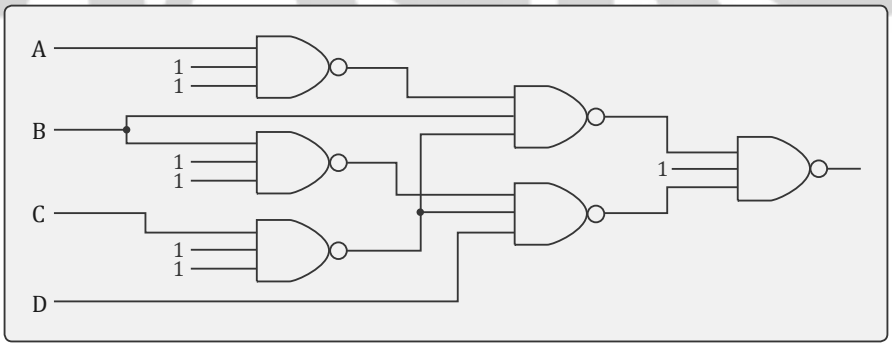
Y su circuito topológico es:

(Dibujar diagrama topológico, mismo que será armado sobre protoboard virtual y físico)

Se puede construir el circuito reducido empleando solo compuertas NO-Y, para lo cual se complementa 2 veces la función y se aplica uno de los complementos, tal como se indica a continuación:

$$Z(A,B,C,D) = (A'BC' + B'C'D)'' = [(A'BC')'\,(B'C'D)']'$$

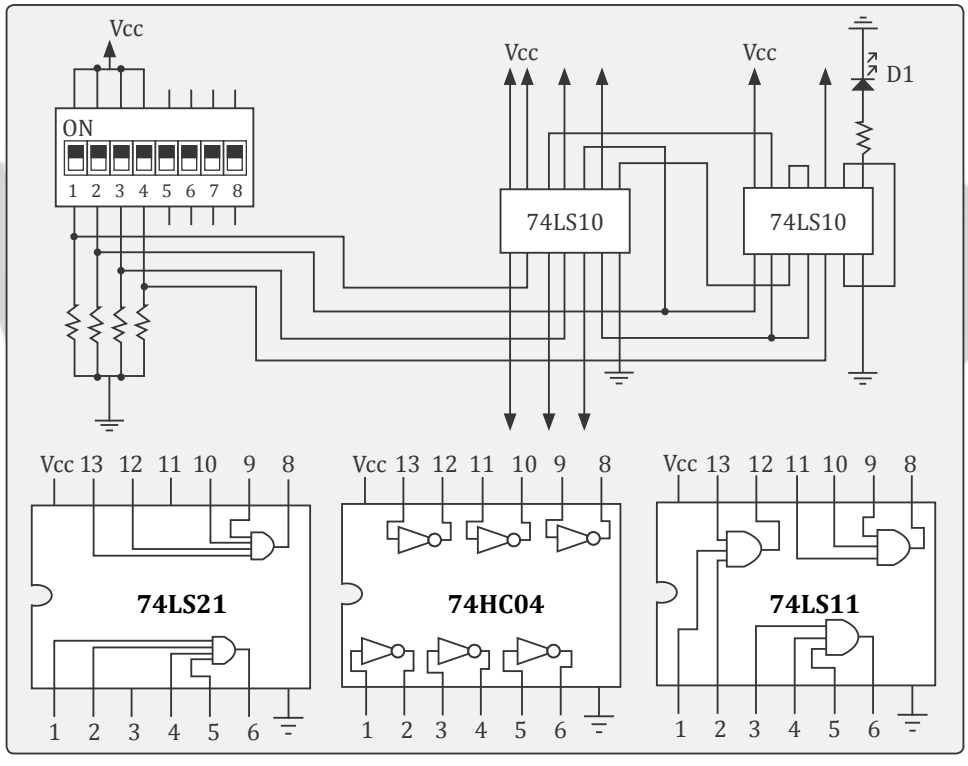
El logigrama para esta función es:



La tabla de verdad es:

A	B	C	D	(A'BC')'	(B'C'D)'	Z
0	0	0	0	1	1	0
0	0	0	1	1	0	1
0	0	1	0	1	1	0
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
0	1	1	1	1	1	0
1	0	0	0	1	1	0
1	0	0	1	1	0	1
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	0

Finalmente, el circuito topológico es:



Procedimiento experimental

1. Arme los tres circuitos topológicos anteriores: el original, el reducido y el realizado solo con compuertas NO-Y.
2. Reporte ventajas y desventajas de la utilización del álgebra de Boole.

_____

_____

_____

_____

3. Como recomendación: los circuitos reducidos y el realizado a base de compuertas NO-Y, ármelo en una misma tablilla de conexiones, utilizando las mismas señales de DIP.

Cuestionario inicial

1. ¿Cuál es el costo del circuito original?

_____

_____

_____

2. ¿Cuál es el costo del circuito reducido?

_____

_____

_____

3. ¿Cuál es el costo del circuito con compuertas NO-Y?

_____

_____

_____

4. ¿Encontró alguna diferencia en la señal de salida de los 3 circuitos anteriores?

_____

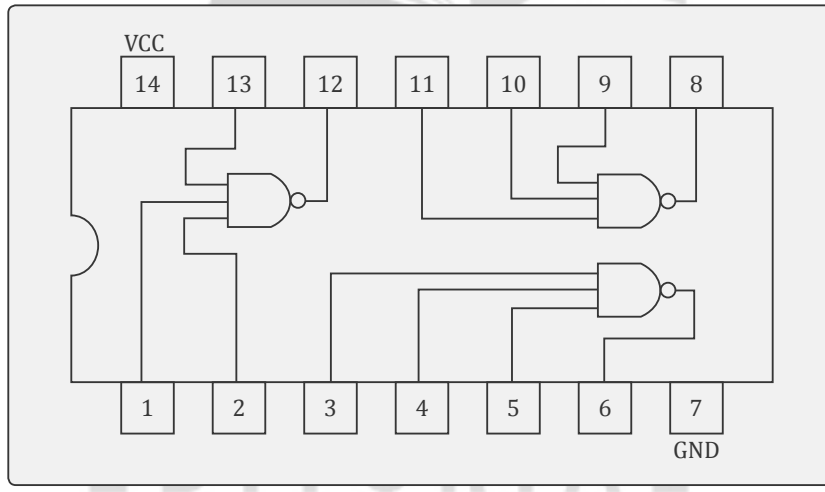
_____

5. Si ocupase alguno de los tres circuitos anteriores, ¿cuál utilizaría? ¿Por qué?

_____

_____

6. Describa el siguiente CI ¿De qué compuerta se trata?



_____

_____

_____

_____

_____

4.1.2 Los mapas de Karnaugh

Los mapas de Karnaugh (M.K) son un método de simplificación de expresiones booleanas por excelencia. Este fue creado por Mauricie Karnaugh, quien formuló una tabla conformada por N número de filas (f) y columnas(c) ordenadas, de modo que puedan alojarse conjuntos de 1 y 0 binarios. Los M.K suelen ser un método matemático bastante ostentoso, aunque, por lo regular, es más efectivo que el álgebra de Boole.

Un mapa de Karnaugh se representa gráficamente de la siguiente manera:

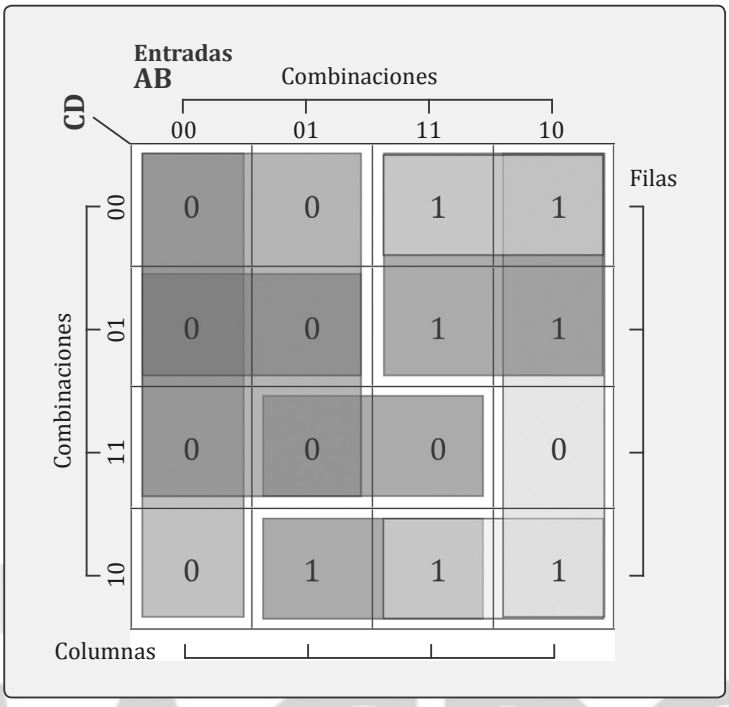


Figura 4.3 Partes que integran un mapa de Karnaugh

Como se puede apreciar en la figura 4.3, este arreglo agrupa una cantidad definida de valores booleanos, con los cuales se conseguirá efectuar una serie de operaciones, que tienen como fin limitar el número de estados para conseguir la reducción de circuitos lógicos. En este ámbito, se describen técnicas de minimización (mini_términos y maxi_términos) que se analizarán más adelante en este apartado. Los mapas de Karnaugh se pueden obtener mediante una tabla de verdad, una expresión lógica, un diagrama lógico y/o un diagrama topológico.

Si se analiza la figura anterior, pueden apreciarse las partes que conforman un mapa de Karnaugh; en la parte inferior se señalan las columnas, mientras que, a la derecha se ubican las filas. Tanto en la parte superior como a la izquierda, se señalan las posibles combinaciones derivadas de una tabla de verdad, previamente formulada. Las entradas (AB|CD) se hallan señaladas en la parte superior izquierda, y repartidas en dos secciones del mapa.

A. Funcionamiento de un M.K

Antes de entrar de lleno a la explicación de este método para la reducción de expresiones booleanas, es necesario remontarse a los conceptos elementales de la lógica binaria y los sistemas digitales, que servirán para comprender el funcionamiento de los mapas de Karnaugh.

a. Niveles de tensión: En electrónica, también conocidos como estados. Su representación es el 1 binario y 0 binario. Siendo 1 el nivel de tensión alto (HIGH), caracterizado por transportar una carga positiva. Por lo general, se esquematiza como un interruptor cerrado, por donde circula la corriente eléctrica. Y el 0 es un nivel de tensión bajo (LOW), caracterizado por el nulo transporte de corriente eléctrica.

b. Tabla de verdad (T.V): Es un arreglo de filas y columnas, estas últimas identificadas como entradas, las cuales se expresan con letras del abecedario (A, B, c, d...Z). En tanto que, sobre las filas se vacían las combinaciones expresadas como cero lógico (0 o nivel LOW) y uno lógico (1 o nivel HIGH).

B. Generación de combinaciones

Una vez dado el número de entradas para una tabla de verdad, es necesario conocer el número de combinaciones (1 y 0) para obtener un resultado de la función (F). La forma de obtener esas combinaciones es mediante el uso de la fórmula de 2 a la N, donde N es el número de columnas de la tabla de verdad (menos la columna de salida F). Ejemplo: Se posee una función de tres entradas representada como F=(A,-B,C). Si se aplica la fórmula 2 a la N $\rightarrow$ 2 a la 3 (porque 3 es el número de entradas en este caso), se obtiene: 2 x 2 x 2 = 8. Finalmente, 8 son el número de combinaciones posibles para una T.V de 3 entradas, como se ve en la siguiente figura:

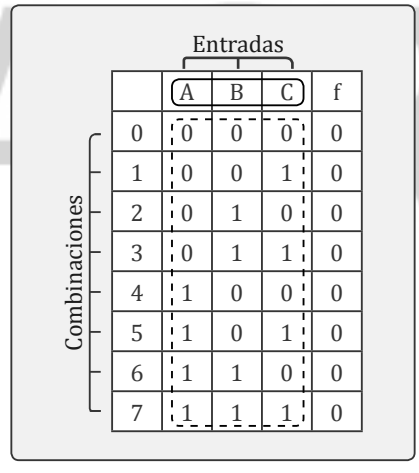


Figura 4.4 Entradas y combinaciones de una tabla de verdad

C. Construcción de un mapa de Karnaugh

Para la creación de un mapa de Karnaugh, que garantice la reducción más apropiada de un circuito, se debe tener a la mano una tabla de verdad, una expresión lógica o un logigrama, aunque lo más común es el uso de tablas de verdad. Con anterioridad, se han expuesto las partes que integran un mapa de Karnaugh, pero ¿qué significan los campos 00, 01, 11, 10 asociados tanto en las filas como en las columnas del mapa? La respuesta es simple, estos números forman parte de un arreglo de números binarios que se encuentran inmersos en una tabla de verdad y que, por lo tanto, dan como resultado las posibles combinaciones (2 a la N). Esas combinaciones se encuentran repartidas en filas y columnas de la siguiente manera[2].

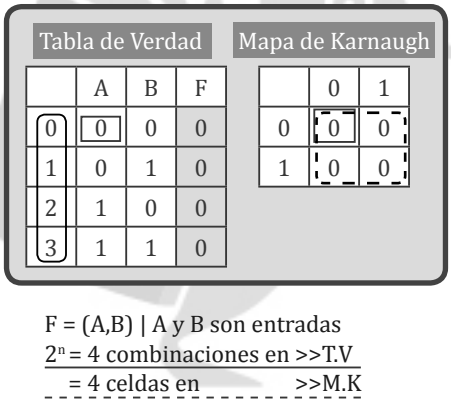


Figura 4.5 Comparación entre una tabla de verdad y un mapa de Karnaugh de dos entradas

La figura anterior muestra, claramente, tanto el trazo de una tabla de verdad de dos entradas (A, B), como su mapa de Karnaugh equivalente. Nótese que la distribución de las combinaciones se encuentra en función de la estructura de la tabla en una posición estándar.

Nota: El M.K tendrá tantas celdas, como número de combinaciones tiene una tabla de verdad, es decir, que si una tabla de verdad posee dos entradas, el número de combinaciones al calcular 2 a la N, será de cuatro, mismo número de celdas que componen al mapa de Karnaugh.

Partiendo de una tabla de verdad puede construirse un mapa de Karnaugh. A continuación, se describe su proceso de elaboración a través de un ejemplo:

[2] Para la obtención de información adicional acerca del tema mapas de Karnaugh, véase Patricia, Q. (2010), «Arquitectura de computadoras».

Ejemplo de elaboración de un mapa de Karnaugh

Dada la siguiente T.V, obtenga su mapa de Karnaugh correspondiente:

	A	B	f
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

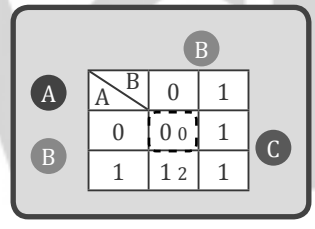
1. Primero identifique el número de entradas en la tabla: F = A,B

   Nótese que el número de entradas en la tabla de verdad son 2, por lo tanto el número de combinaciones son 4.

2. Después, identifique el número de combinaciones, haciendo uso de la fórmula 2 a la N.
3. Identifique el valor de salida de la función F en la tabla (arrojando valores de 0 o 1).
4. Dadas las combinaciones 00, 01, 10 y 11 en la tabla, identifique en qué posición se encuentra presente el valor 1 y 0. Para este ejemplo, la distribución se aprecia de la siguiente manera:

   00 = 0 | 01 = 1 | 10 = 1 y 11 = 1

5. Diseñe una tabla como se muestra a continuación:



   a. Se reparten las entradas en esta nueva tabla.
   b. Se reparten las posibles combinaciones.
   c. Se vacían los valores 1 y 0 en la posición correspondiente de la tabla.

D. POS y SOP para un mapa de Karnaugh

POS y SOP son dos formas de expresar una función booleana que desea simplificarse.

SOP (*sum of product*) o suma de productos. Ejemplo: (A.B) + (B.C).

POS (*product of sum*) o producto de sumas. Ejemplo: (A+B) (B+C).

Ahora, ¿tienen alguna relación los niveles de tensión 1 y 0 con el método SOP y POS? La variable de salida F en una tabla de verdad puede tomar el valor 1 o 0.

Cuando se desea simplificar una expresión booleana y desea tomarse en cuenta la combinación cuyo valor sea 1, se tiene presente un método SOP.

Cuando se desea simplificar una expresión booleana y desea tomarse en cuenta la combinación cuyo valor sea 0, se tiene presente un método POS.

¿Qué es un minitérmino y un maxitérmino?

En resumen, cuando se aplica un método SOP se tiene presente una simplificación por minitérminos (*minterms*). Cuando se aplica un método POS se alude a una simplificación por maxitérminos (*maxterms*).

Minterms = SOP = Reducción por 1

Maxterms = POS = Reducción por 0

Resumiendo, toda simplificación de funciones requiere ser expresada como POS o SOP.

Si se elige una suma de productos (SOP), la simplificación quedará expresada en minitérminos.

Si se elige un producto de sumas (POS), la simplificación está expresada en maxitérminos.

Obtención de una expresión booleana dada una tabla de verdad

Para obtener una expresión booleana de la T.V. es necesario tomar las combinaciones cuyo valor de salida (F) de la tabla sea 1 (ya se ha elegido con anterioridad la simplificación por minitérminos).

a. Se listan las combinaciones en 1, sustituyendo los ceros y los unos por las variables de entrada: para ello se niega la variable cuyo valor sea cero, y dejando la variable sin negación cuando el valor sea uno, como se ve a continuación:

	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

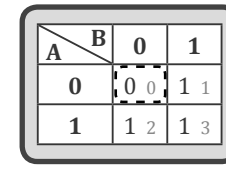
01 = A' B $\rightarrow$ F=1

10 = A B' $\rightarrow$ F=1

11 = A B $\rightarrow$ F=1

b. Se forma la expresión según el método elegido (SOP): se procede a estructurar la función de acuerdo al resultado obtenido en la tabla (en este caso se obtienen tres términos en función del número de 1 lógicos resultantes).

$$F = A'B + AB' + AB$$



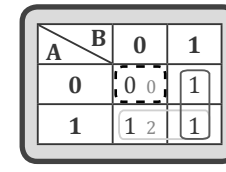
Esta expresión es la que se tiene que simplificar mediante el uso de M.K. (nótese la distribución de estados en su posición correspondiente).

Pasos para la reducción por SOP

Para este momento ya se cuenta con una T.V, un M.K y una expresión booleana a simplificar por el método SOP, se procede ahora al proceso de simplificación:

1. Generación de conjuntos

a. Agrupar las casillas cuyo valor sea 1 (potencias de 2). La idea consiste en crear conjuntos de 1 sin dejar ningún valor sin considerar.



2 conjuntos distintos

Nótese que se han conformado dos conjuntos de 1, lo que implica una reducción de términos de la función F = A'B + AB' + AB. Por lo tanto se obtiene: A/B = 1 0, 11 para el primer conjunto (agrupación horizontal) y A/B = 0 1, 11 para el segundo conjunto (agrupación vertical).

Nota: Entre menor número de conjuntos se formen y entre más grandes sean, mejor será la simplificación de la función.

2. Uso de la técnica de cancelación

Esta técnica permite la reducción, tomando como base las entradas del mapa. Las cuales, son sustituidas por un 0 o un 1 lógico según corresponda. De este modo, resulta más sencillo trabajar con la construcción de términos de la expresión booleana (la cual se expresa como SOP o POS). Antes de citar algunos ejemplos y aplicar esta técnica, se recomienda analizar el siguiente contenido:

Siendo A y B (00, 01, 10, 11)
0 con 1 = cancelación
1 con 0 = cancelación
1 con 1 = A B
0 con 0 = A' B'

- **¿En qué consiste la técnica de cancelación?**

  Partiendo de la existencia de dos variables de entrada A y B, con una combinación de 1 y 0 asociada, se cancelarán las combinaciones que sean distintas entre sí, cuando ambas entradas sean 1, el valor de salida será 1; y si se trata de una combinación en la que ambas entradas sean 0, el resultado será 0. Finalmente, hay que sustituir el 0 binario por la variable de entrada, pero invertida; cuando el valor sea 1 binario, la entrada correspondiente no se invierte y pasa sin ser alterada, tal y como lo muestra la siguiente figura:

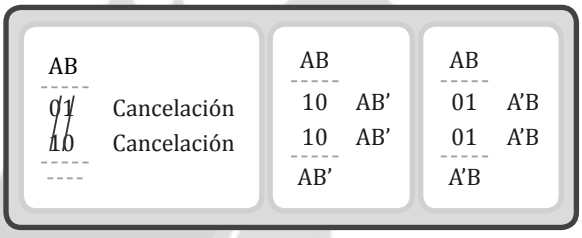


Figura 4.6 Obtención de términos mediante el método de cancelación

- Como se ha mencionado, al seguir este principio se obtiene la construcción de términos para la expresión (función reducida). Los resultados obtenidos para el ejemplo anterior, se muestran a continuación:

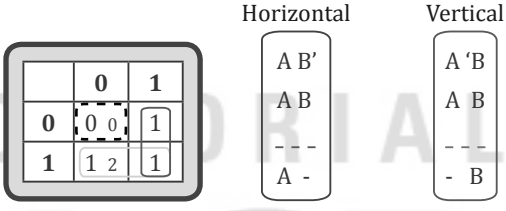


Figura 4.7 Agrupación y generación de resultados

Para el primer conjunto = A, para el segundo conjunto = B

El resultado de la función F, será igual a la suma de A + B quedando como:

F = (A + B)

La función original = F = A'B + AB' + AB

La reducción por SOP: F= A + B

> **Nota:** Actualmente, se pueden implementar mapas de Karnaugh de 2, 3, 4, 5 y hasta 6 entradas.

Construcción de un mapa de Karnaugh partiendo de una expresión booleana

Ahora partiendo de una expresión booleana puede construirse un mapa de Karnaugh. A continuación, se describe su proceso de elaboración:

Ejemplo de construcción de un mapa partiendo de una expresión lógica

Dada la siguiente expresión booleana, obtenga su mapa de Karnaugh correspondiente:

$$F = A\,B + A'\,B' + A'\,B + A\,B$$

1. Se crea una tabla de verdad, según el número de entradas en la expresión. Nótese que para este caso son 2 (A y B).

	A	B	F
0	0	0	X
1	0	1	X
2	1	0	X
3	1	1	X

2. Tome nota de la primera combinación (0 0) y sustituya el valor para A y B en la expresión dada:

F = A B + A' B' + A' B + A B
F = 0 0 + 1 1 + 1 0 + 0 0
F = 0 + 1 + 0 + 0
F = 1, por lo tanto, la primera combinación es 1.

	A	B	F
0	0	0	1
1	0	1	X
2	1	0	X
3	1	1	X

Recuerde: Si el valor 1 se invierte, pasa a 0, y si el 0 se invierte pasa a 1 lógico.

3. Continúe sustituyendo los valores de las combinaciones posteriores sobre la expresión booleana.

	A	B	F
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	1

4. Una vez obtenida la tabla de verdad, vacíe los datos en el M.K como se explicó en la sección de construcción de un mapa de Karnaugh, partiendo de una T.V.

AB	0	1
0	1	1
1	0	1

F = A' + B

Resultado obtenido:
La función original: F = A B + A' B' + A' B + A B
La reducción por SOP: F = A' + B

Un mapa de Karnaugh de 3, 4, 5 y 6 entradas se esquematiza de la siguiente forma:

1. Mapa de 3 entradas

	00	01	11	10
0	0 0	0 1	0 3	0 2
1	0 4	0 5	0 7	0 6

2. Mapa de 4 entradas

	00	01	11	10
00	0 0	0 1	0 3	0 2
01	0 4	0 5	0 7	0 6
11	0 12	0 13	0 15	0 14
10	0 8	0 9	0 11	0 10

3. Mapa de 5 entradas

	000	001	011	010	100	101	111	110
00	0 0	0 1	0 3	0 2	0 16	0 17	0 19	0 18
01	0 4	0 5	0 7	0 6	0 20	0 21	0 23	0 22
11	0 12	0 13	0 15	0 14	0 28	0 29	0 31	0 30
10	0 8	0 9	0 11	0 10	0 24	0 25	0 27	0 26

4. Mapa de 6 entradas

	000	001	011	010	100	101	111	110
000	0 0	0 1	0 3	0 2	0 16	0 17	0 19	0 18
001	0 4	0 5	0 7	0 6	0 20	0 21	0 23	0 22
011	0 12	0 13	0 15	0 14	0 28	0 29	0 31	0 30
010	0 8	0 9	0 11	0 10	0 24	0 25	0 27	0 26
100	0 32	0 33	0 35	0 34	0 48	0 49	0 51	0 50
101	0 36	0 37	0 39	0 38	0 52	0 53	0 55	0 54
111	0 44	0 45	0 47	0 46	0 60	0 61	0 63	0 62
110	0 40	0 41	0 43	0 42	0 56	0 57	0 59	0 58

Para comprobar el funcionamiento de un mapa de Karnaugh se puede hacer uso de herramientas de software como *Logisim*, *Karnaugh Map Minimizer*, *Karnaugh Analyzer*.

En este capítulo se han revisado los métodos utilizados en la electrónica digital para la reducción de circuitos electrónicos. Se han incluido los teoremas y postulados del álgebra de Boole y el procedimiento para la obtención de reducciones mediante mapas de Karnaugh. En el siguiente capítulo se darán a conocer las herramientas físicas idóneas para la construcción de circuitos, desarrollo de proyectos e implementación de soluciones electrónicas.

Práctica de laboratorio n.º 4

Minimización

Objetivo de la práctica: Comprobar mediante Mapas de Karnaugh una serie de ejercicios prácticos de minimización aplicados a la ingeniería de hardware.

Duración: 2 horas.

Materiales:

- El necesario para cablear los ejercicios.
- Simulador.

PARTE 1

1. Simplifique por Karnaugh la función cuya expresión en términos canónicos es:

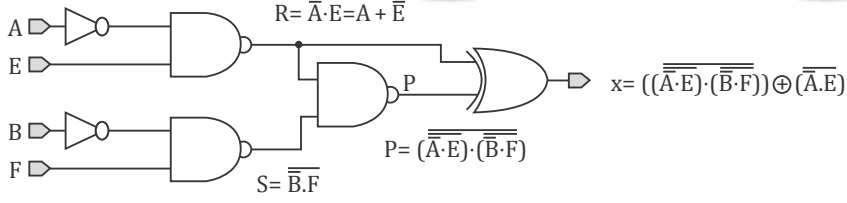
$$f(x, y, z) = \sum_{3} m(3, 5, 6)$$

2. Utilizando los mapas de Karnaugh, simplifique las siguientes funciones de conmutación, obtenerlas en función de suma de productos o producto de sumas:
   a. f(w, x, y, z) = ∑m(5, 6, 9, 10)
   b. f(x, y, z) = ∑m(2, 3, 4, 5, 6, 7)
   c. f(x, y, z) = ∑m(2, 4, 5, 6)
   d. f(w, x, y, z) = ∑m(3, 6, 7, 11, 12, 14, 15)

Una vez hecho lo anterior, cablee sobre protoboard virtual y físico (original + simplificado). Posteriormente compruebe su igualdad.

PARTE 2

1. Para las siguientes funciones lógicas obtenga:



   a. Elabore la tabla de verdad del circuito.
   b. Haga el mapa de Karnaugh.
   c. Obtenga, a partir del mapa de Karnaugh, la ecuación simplificada en suma de productos (SOP) y productos de sumas (POS).

PARTE 3

Realice los siguientes ejercicios:

1. Detector de números primos en BCD-XS3

Se quiere realizar un circuito que reciba un número BCD-XS3 (4 bits) y devuelva '1', si el número recibido es primo y devuelva '0' si no lo es.

Se considerará el número 1 como número primo. El cero no es un número primo. En ningún caso el circuito recibirá números que no estén codificados en BCD-XS3.

Se pide:

a. Realizar la tabla de verdad de la señal de salida.

b. Obtener la expresión reducida en suma de productos, y producto de sumas

c. Dibujar el esquema en puertas de estas expresiones

2. Alarma

Se quiere realizar un circuito para activar la alarma de incendios (A) para la evacuación de un edificio. Para ello se tiene un sensor de gases (G), un sensor de humos (H) y dos señales procedentes de un termómetro que indican si la temperatura es mayor de 45 ºC (T45), si la temperatura es mayor de 60 ºC (T60).

Debido a que a veces los sensores detectan humos y gases que no siempre proceden de incendios (por ejemplo de los cigarrillos o las cocinas), para evitar falsas alarmas, la señal A se activará cuando se cumplan las siguientes condiciones:

- Si la temperatura es mayor de 60 °C siempre se activará la alarma
- Si la temperatura está entre 45 °C y 60 °C se activará la alarma solo si han detectado gases o humos (o ambos).
- Si la temperatura es menor de 45 °C se activará la alarma solo si se detectan gases y humos.

Resumiendo, las 4 señales binarias de entrada y la salida:

- G: vale '1' si se detecta GAS resultante de la combustión.
- H: vale '1' si se detecta HUMO.
- T45: vale '1' si la temperatura es superior a 45 °C
- T60: vale '1' si la temperatura es superior a 60 °C

La señal de salida A (alarma) se activará a nivel alto.

a. Realice la tabla de verdad de la señal de alarma (A) a partir de las señales de entrada (G, H, T45, T60).
Explicarla brevemente.

b. Obtenga la expresión reducida en suma de productos y producto de sumas
c. Dibuje el esquema en puertas de estas expresiones.
d. Documente todos los ejercicios (EXPOSICIÓN) fin de la práctica.

Práctica de laboratorio n.º 5

Mapas de Karnaugh

Objetivo de la práctica: Comprobar la importancia de los mapas de Karnaugh en la minimización de funciones de conmutación basándose en la suma de productos.

Duración: 4 horas.

Materiales:

Será planteado por el alumno, de acuerdo a las especificaciones de la práctica. Debe listarlo a continuación:

_____

_____

_____

_____

Procedimiento:

Problema: Un circuito lógico combinatorio recibe dos números de tres bits cada uno, A = A2, A1, A0; y B = B2, B1, B0.

- ✔ Diseñe un circuito mínimo de suma de productos para producir una salida f = 1 siempre que A sea mayor que B.

Solución: Tomando en cuenta todas las combinaciones de los dos números de tres bits y las condiciones del problema, se realiza la siguiente tabla funcional:

DEC	A			B			f
	A_2	A_1	A_0	B_2	B_1	B_0	
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	0

Hasta 63 base 10.

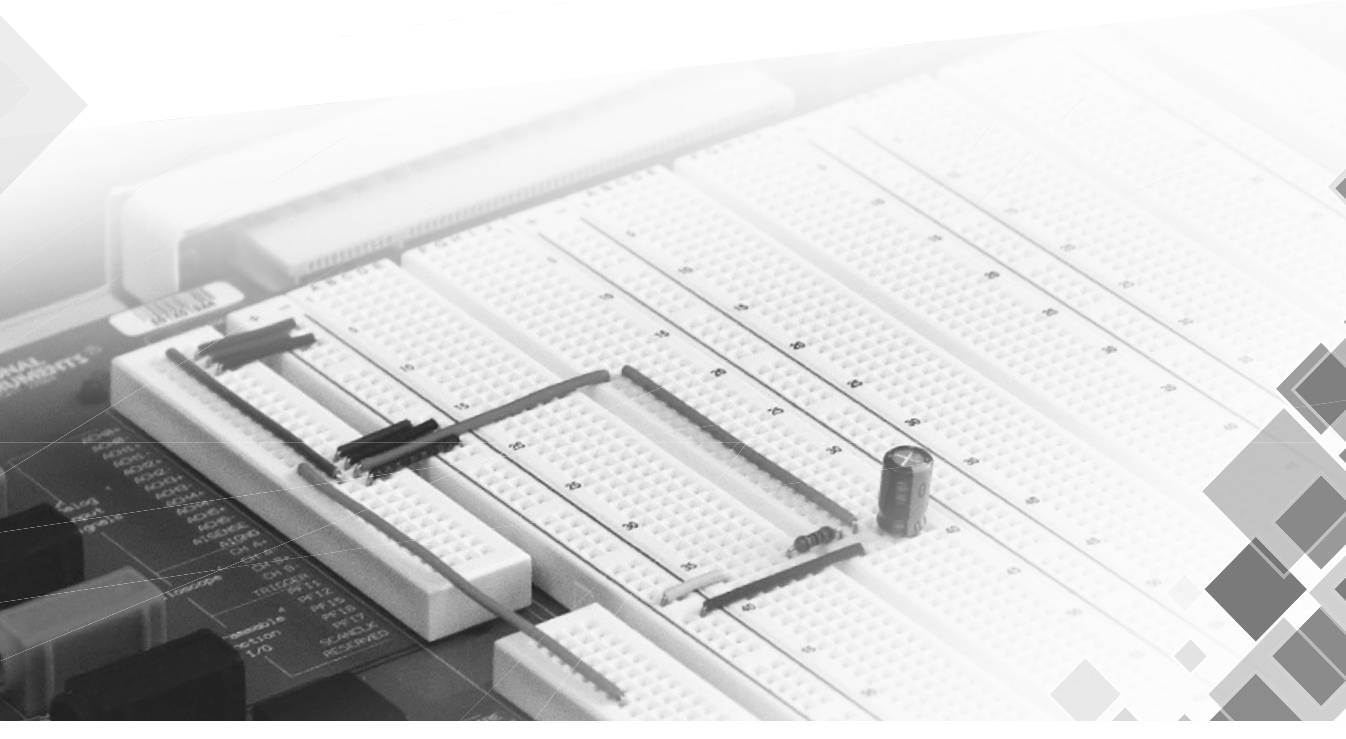
De la tabla funcional, se obtiene la función de salida f en forma canónica (A2, A1, A0, B2, B1, B0) = SUMA minitérminos (8, 16, 17, 24-26, 32-35, 40-44, 48-53, 56-62).

Realizar:

a. Mapa de Karnaugh correspondiente.
b. Función mínima.
c. Logigrama de la función reducida.
d. Diagrama topológico.
e. Cablear el circuito con base en el diagrama del inciso d.

Capítulo 5

Componentes electrónicos y herramientas de montaje



EDITORIAL
MACRO®

Las placas de prototipado son un tipo de herramientas de montaje, regularmente utilizadas en el campo de la electrónica, las cuales han sido diseñadas para montar y probar prototipos electrónicos. Hay de diferentes tamaños, arquitecturas y fabricantes. Son consideradas como elementos pasivos o conductores que permiten la inserción y colocación de componentes electrónicos utilizados para el montaje de un circuito.

Con el paso del tiempo, las placas de prueba han tenido una evolución, lo que antes era un elemento pasivo, hoy se ha convertido en una *suite* de instrumentos usados comúnmente en un solo formato compacto. Esto con el fin de hacer más sencillo el aprendizaje autodidacta del usuario y la interacción con proyectos reales. Una de las herramientas de montaje más comerciales es la llamada NI ELVIS (*NI Educational Laboratory Virtual Instrumentation Suite*) de la compañía National Instruments. Consiste en una *suite* orientada a la educación y, originalmente, pensada para usuarios que desean introducirse al ámbito de la electrónica, la ingeniería de hardware, la mecatrónica, la automatización, la robótica, la tecnología, etc. Contiene diversos componentes como un osciloscopio, multímetro digital, generador de funciones, analizador de señal dinámica, un área para montaje, interfaces de conexión a la PC a través de distintos puertos.

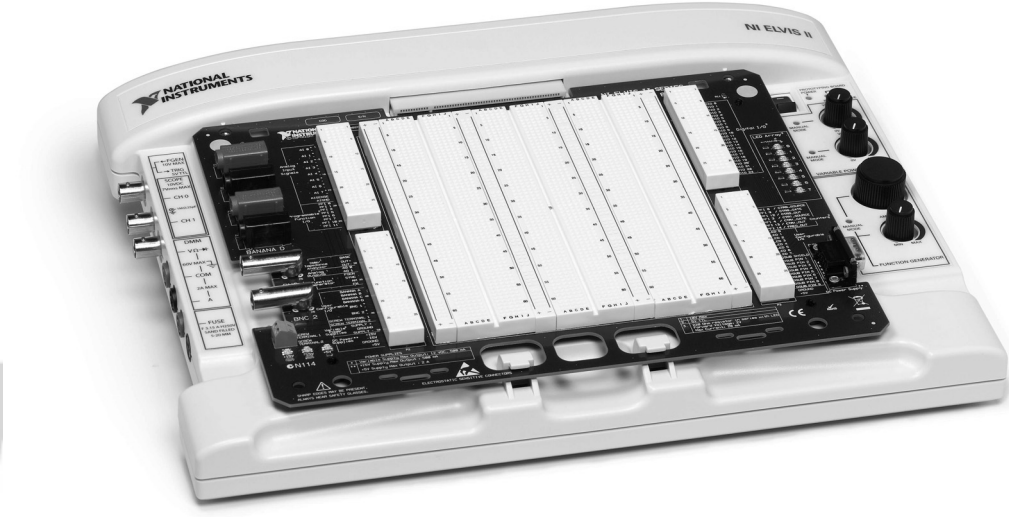


Figura 5.1 NI ELVIS, herramienta que permite al usuario el desarrollo de proyectos electrónicos orientados al desarrollo y prueba de hardware

5.1 BREADBOARD

El protoboard o breadboard (tablero para montaje de prototipos) permite colocar y retirar componentes electrónicos las veces que sea necesario. Es un elemento conformado por líneas de contactos distribuidas tanto en filas como en columnas. Por lo general, se conforman por varias partes, las cuales se muestran en la siguiente figura:

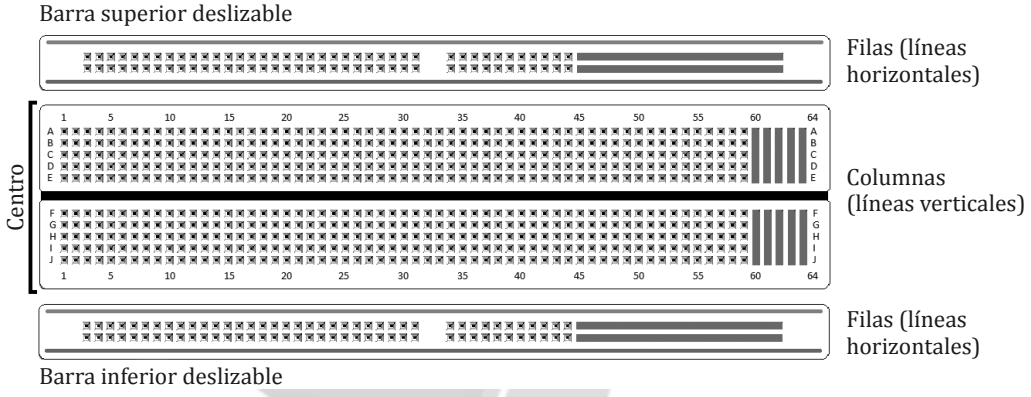


Figura 5.2 Esquema que ilustra las partes y la distribución de líneas de contacto de un protoboard
Fuente: el autor.

Un protoboard se encuentra rotulado con dos líneas horizontales, una de color azul y otra roja, ambas situadas tanto en la parte superior como en la parte inferior. Estas líneas, a menudo, son utilizadas para la polarización de circuitos (la línea azul se encuentra asociada a GND y la línea roja a VCC). La adecuada manipulación de este elemento garantiza una conexión segura e impide que se quemen ciertos componentes.

El protoboard contiene delgadas láminas en su interior (que se pueden apreciar desde su base) distribuidas de manera horizontal y vertical formando filas y columnas, respectivamente. Como se puede apreciar en la figura, las líneas dedicadas a VCC y GND están trazadas en forma horizontal.

Las líneas verticales o columnas están distribuidas en el centro del protoboard. Estas, constantemente, sirven para montar componentes como circuitos integrados, *displays*, microcontroladores, resistencias, transistores y demás elementos.

Generalmente, existen diferentes tipos de diseño de placas para montar prototipos, entre los cuales destacan el protoboard de plástico, que permite montar y desmontar componentes y el protoboard soldable. Este último, a diferencia del protoboard de plástico, permite soldar sobre su base los diferentes elementos electrónicos, evitando así que se caigan o extravíen. El breadboard soldable es idéntico al de plástico en cuanto a la distribución de sus terminales de conexión. Actualmente, en el mercado electrónico se pueden conseguir dos formatos especiales de protoboard soldable: el perfboard y el stripboard; ambas consisten en placas de cobre, cuya distribución de orificios se encuentra trazada en función del prototipo electrónico. Por su parte, el perfboard mantiene una distribución de orificios de cobre que no están interconectados entre sí. Este tipo de placas requieren que cada componente esté soldado a su superficie y, además, que las interconexiones entre ellos se realicen a través de cables o caminos de soldadura.

El stripboard es un tipo especial de perfboard con patrón en donde los orificios se interconectan formando filas de material conductor.

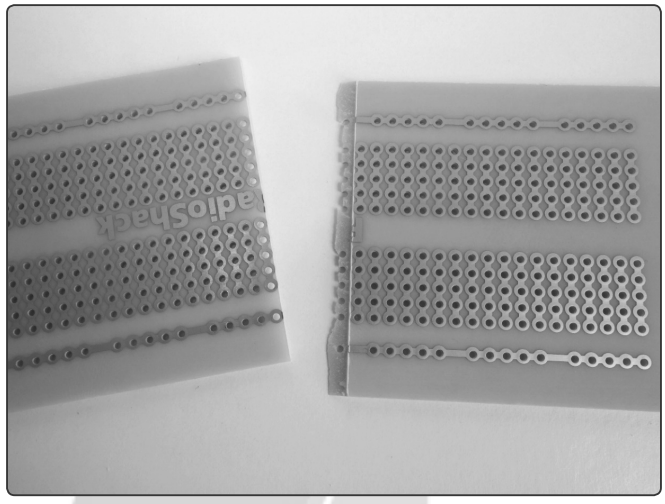


Figura 5.3 Stripboard, un tipo especial de placa para el montaje de prototipos electrónicos

5.2 SIMULADORES

Los simuladores son herramientas que facilitan la tarea de montaje de circuitos de manera física. Gracias a estas utilerías resulta sencillo comprobar resultados, depurar errores y trabajar sin riego de quemar o echar a perder componentes. Actualmente, son muchas las herramientas que se alojan en la Web, por lo que siempre resulta interesante conocer tanto sus pros, como sus contras. Más adelante se hace la recomendación de algunas de estas herramientas.

5.2.1 Protoboard virtual

Existen diferentes herramientas de software que permiten simular la conexión de algún circuito o diseño electrónico mediante placas virtuales. Estas se pueden descargar de Internet e incluso utilizarlas vía *online*. Uno de los simuladores de protoboard más populares es el llamado constructor virtual y simulador de circuitos digitales. Otra atractiva opción se trata de *WinBreadboard*.

La finalidad de utilizar alguna herramienta de software como las ya mencionadas es evitar dañar elementos electrónicos ante una inadecuada conexión. Permiten la evaluación y prueba de funcionamiento del montaje antes de su presentación en diseño físico. Estas herramientas de simulación son muy fáciles de comprender, aunque requieren de destreza y lógica al momento de cablear. Para conocer su interfaz visual y comprender su uso, se recomienda atender la siguiente infografía.

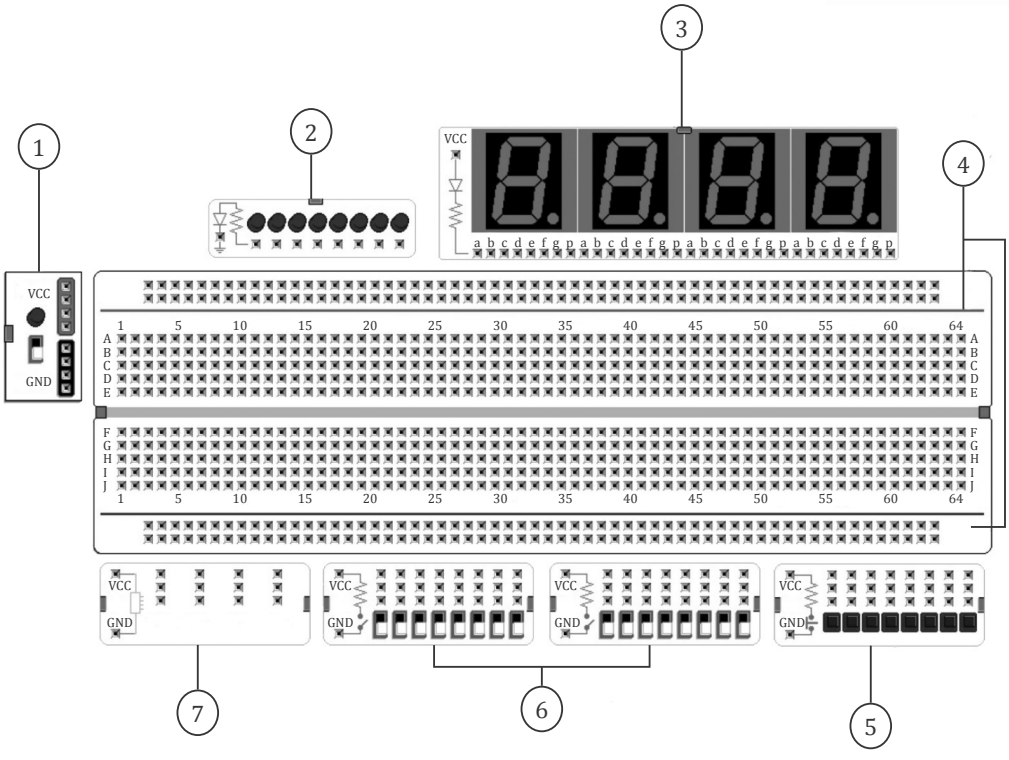


Figura 5.4 Infografía que muestra uno de los simuladores de protoboard más populares

N.º	Componente	Descripción
1	Fuente de alimentación	Se trata de un módulo de polarización. Es el equivalente a una batería (que puede variar entre 3.3 V, 5 V, o 9 V).
2	Línea de led	Son barras de led que se pueden agregar al área de trabajo. En este simulador se encuentran distribuidas de manera vertical u horizontal, y los hay de color verde, amarillo y rojo. Generalmente, representan las salidas de un circuito.
3	*Display* de 7 segmentos	Este simulador dispone de hasta 4 *display* de 7 segmentos cada uno. Cuenta, además, con la opción de conectarlos como ánodo o cátodo común. A través de estos se visualizan números del 0 al 9. Muy útiles al momento de trabajar con contadores, relojes, conversores de binario a decimal o hexadecimal.
4	Líneas indicadoras de polaridad	En el protoboard virtual, de igual modo se han incorporado las líneas que corresponde a VCC Y GND. En el simulador se identifican solo una línea azul y una roja.
5	Pulsadores	Son los equivalentes a *pushbotton* o botones de pulsación. Muy útiles cuando se desean simular teclados matriciales.
6	DIP Switch	Son interruptores de dos posiciones (0 y 1). Los cuales a su vez son tratados como entradas.
7	Relojes de frecuencia	Permiten regular la frecuencia (de 1 a 10Hz). Muy útiles en contadores, temporizadores (como el tradicional 555).

Como se puede apreciar, la interfaz gráfica que ofrece esta utilería es muy intuitiva, ya que, incluye, además de los componentes necesarios para el montaje de proyectos, un tutorial de uso de componentes electrónicos.

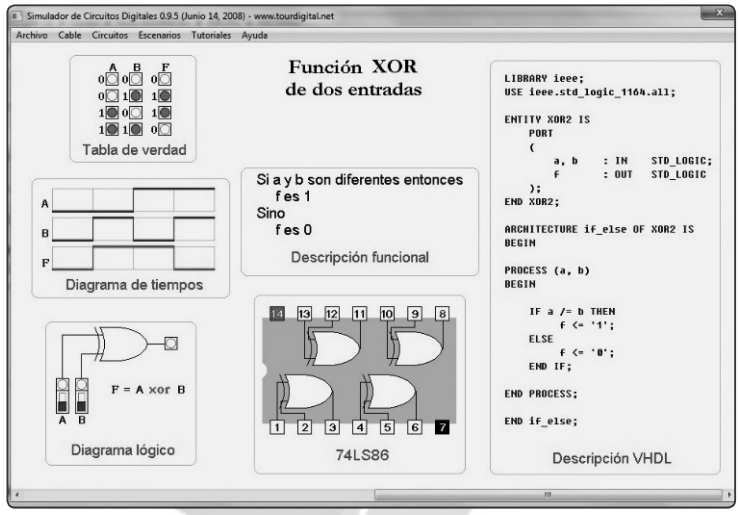


Figura 5.5 Integración de tutoriales que ilustran el funcionamiento de un CI a través del constructor virtual de circuitos

5.3 COMPONENTES ELECTRÓNICOS

Hasta ahora se ha descrito, de manera breve, uno de los componentes electrónicos de mayor auge en la electrónica digital, el circuito integrado. Sin embargo, es el turno para describir algunos otros componentes electrónicos de alta demanda en el mercado de la electrónica y la ingeniería de hardware. En el capítulo uno de este libro se puede apreciar la clasificación de estos elementos y su representación. Aunque, en esta ocasión, se pretende hacer mayor énfasis en su constitución y funcionamiento.

Un componente electrónico es aquel elemento que, generalmente, forma parte de un circuito electrónico. Estas piezas suelen estar encapsuladas o cubiertas de algún material como cerámica, metal o plástico. Poseen cuando menos dos terminales o patillas metálicas asociadas a una línea de VCC y GND, respectivamente. Tienen una función específica y actúan de acuerdo a un diseño lógico.

En esta sección se describen los componentes electrónicos más utilizados. Lo cuales, van a servir de guía al usuario al momento de enfrentarse con el diseño de circuitos, manejo de plataformas o creación de proyectos.

a. Resistencia: Es un elemento pasivo cuya función es oponerse al paso de la corriente de un punto a otro. El material de fabricación, generalmente, es de carbón con un recubrimiento de óxido de metal o carbón depositado. Su unidad de medida se expresa en ohmios. Son de distintos tamaños y disposiciones (por ejemplo las resistencias que se sueldan sobre PCB, son muy pequeñas e incluyen tres dígitos donde se rotulan su valor y el multiplicador) y no poseen indicador de polaridad (son unipolares, es decir, que no importa el sentido en que se conectan). Una resis-

tencia maneja por lo general dos valores de tolerancia o precisión (+-5 % y +-10 %). Por ejemplo, una resistencia de 220 ohmios puede presentar un valor aproximado de 198 ohmios o 242 ohmios para el 10 %.

Si se desea profundizar un poco más al respecto, puede consultar el capítulo uno de este libro en el tema código de resistencias.

b. Diodo: Se trata de un componente electrónico integrado por dos terminales que permite el flujo de corriente en un solo sentido. Los diodos, por lo general, se pueden conectar de acuerdo a la polarización deseada (directa o inversa), y es, comúnmente, empleado como rectificador (conversión de una corriente alterna a continua).

Su uso se limita a proteger un circuito ante una inadecuada conexión de componentes electrónicos (polaridad al revés) que pudiera ocasionar un daño total o parcial a un circuito.

Uno de los diodos más conocidos y empleados en la electrónica es el diodo emisor de luz o led (*light emitting diode*). Posee una terminal positiva (terminal más larga llamada ánodo) y una negativa (terminal más corta llamada cátodo), emite luz de acuerdo a la intensidad de corriente proporcionada y existe una alta gama de tamaños, tipos y colores. Habitualmente, la intensidad más idónea para el funcionamiento óptimo de un led es de 15 mA. El voltaje más apropiado para su brillo puede variar según el color del led, va de 3 V a 3,6 V para el ultravioleta (UV), el blanco o azul, una tensión de 2,5 V a 3 V para el verde, 1,9 V a 2,4 V para el rojo, naranja, amarillo y de 1 V a 1,5 para el infrarrojo.

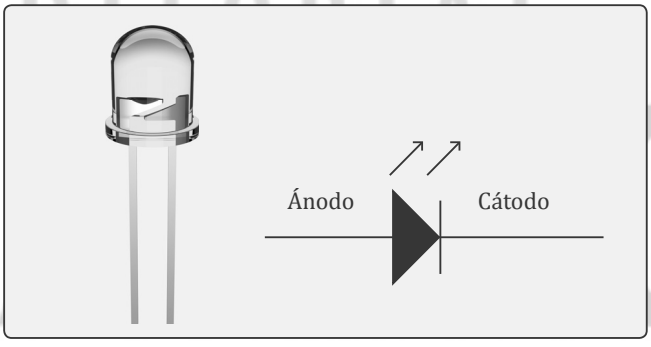


Figura 5.6 Los ledes, claro ejemplo de diodo, funcionan mediante una polarización directa

c. Transistor: Es un componente electrónico que se conforma de tres terminales: colector (C), emisor (E) y base (B), generalmente llamado triodo. Su función consiste en restringir o permitir el flujo de corriente eléctrica entre dos contactos (emisor y colector) según la presencia o ausencia de corriente en un tercer contacto (B). Puede entenderse como una resistencia variable entre dos puntos, cuyo valor es controlado mediante la aplicación de una determinada

corriente sobre un tercer punto. Este componente se puede conseguir en diferentes tamaños y tipos.

Los transistores suelen clasificarse según su tecnología de fabricación y funcionamiento, los transistores de tipo bipolar o BJT (*bipolar junction transistor*) y los transistores de tipo efecto de campo o FET (*field effect transistor*). Es habitual el uso de transistores PNP (positivo negativo positivo) y NPN (negativo positivo negativo).

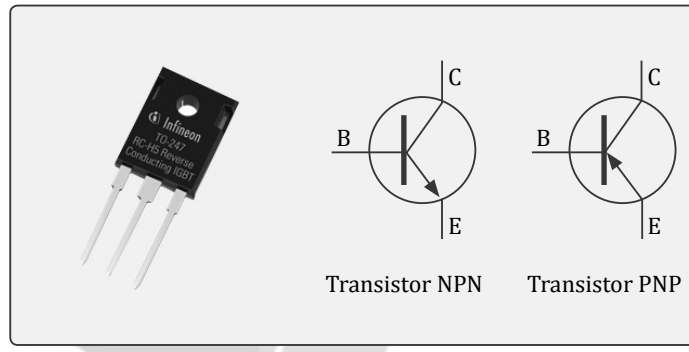


Figura 5.7 El transistor contiene tres terminales: colector, emisor y base

d. Potenciómetro: Se denomina como resistencia de valor variable, pues, internamente, es muy parecida a la resistencia tradicional, solo que esta incluye tres contactos. Los extremos se conectan a VCC y GND, respectivamente sin importar el orden de polarización (por cada extremo se cuenta con un valor máximo fijo de resistencia, dejando una parte de esos valores en el contacto central), y se deja la terminal del centro para la salida (*vout*) de datos. El potenciómetro es, originalmente, un componente analógico, sin embargo, actualmente, es posible encontrarlos en su formato digital. Es muy utilizado en la electrónica, sobre todo en amplificadores de audio, reguladores de temperatura, algunas fuentes de alimentación y en muchos sistemas industriales. Con este elemento es sencillo alterar el paso de la corriente en un circuito, para ello es necesario no dejar de lado la conocida ley de Ohm. Para la obtención de cálculos en la resistencia máxima de un potenciómetro entre sus dos extremos se requiere de la suma de las resistencias presentes entre los extremos y el contacto central (R1 y R2, respectivamente).

Actualmente, es muy común encontrar algún tipo de potenciómetro que opere en el interior de un reproductor de CD/DVD/BR para PC. Es más pequeño que el tradicional y permite hacer una calibración de dicha lectora. La unidad de medida de un potenciómetro es el Ohmio.

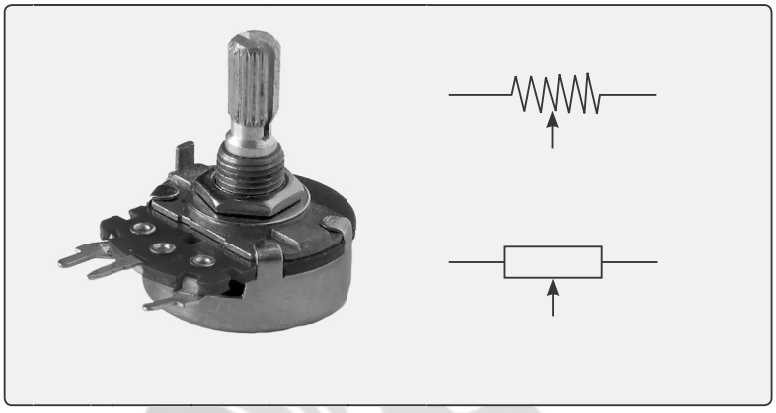


Figura 5.8 Los potenciómetros, adquiridos de 5 kilo ohmios hasta 500 kilo ohmios. A la derecha se muestra su símbolo

e. Fotorresistencia: Se trata de un tipo de resistencias de valor variable. Es también conocida como LDR (*light dependent resistor*) o, simplemente, celda CdS (sulfuro de cadmio). Este elemento se caracteriza porque su valor varía de acuerdo a la cantidad de luz que incide en él (y que a veces es catalogado como sensor). Estas poseen dos terminales o patillas, un elemento fotoresistivo y una carcasa.

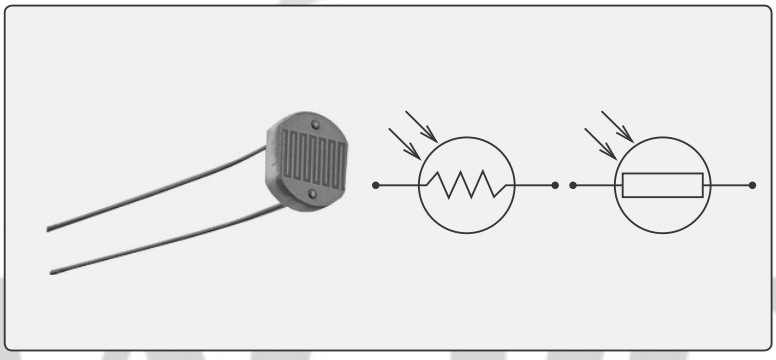


Figura 5.9 La fotorresistencia, elemento que al igual que la tradicional resistencia de carbón, es unipolar. A la derecha se muestra su símbolo

f. Condensador: Se conoce también como capacitor. Su función es almacenar carga eléctrica en cantidades limitadas como si se tratara de una fuente de alimentación adicional. Estos elementos pueden ser unipolares o no y, generalmente, se clasifican de acuerdo al material de fabricación, cerámico, de aire, de papel, electrolítico, condensador variable, etc.

En realidad, los condensadores son usados en multitud de aplicaciones como baterías y memorias por su cualidad de almacenar carga, para realizar descargas rápidas (como la luz flash de una cámara fotográfica), para mantener corrientes estables (como las generadas por un rectificador), para evitar caídas de corriente puntuales en los circuitos, es decir, la función de bypass, etc.

La capacidad (C) de un condensador (expresada en faradios) es su característica más importante y se puede definir como la relación (normalmente, de un valor constante) que existe entre la cantidad de carga eléctrica (Q) que almacena en un momento dado y la tensión (V) aplicada en el momento. Concretamente, se define como: **C = Q/V**. La manipulación de un capacitor debe ser de extrema precaución, pues son susceptibles a estallar.

A continuación, se aprecia la unidad de medida y su equivalencia:

1 µF= 10 a la menos 6 F = 0 000001

1 ηF= 10 a la menos 9 F = 0 000000001

1 ρF= 10 a la menos 12 F = 0 000000000001

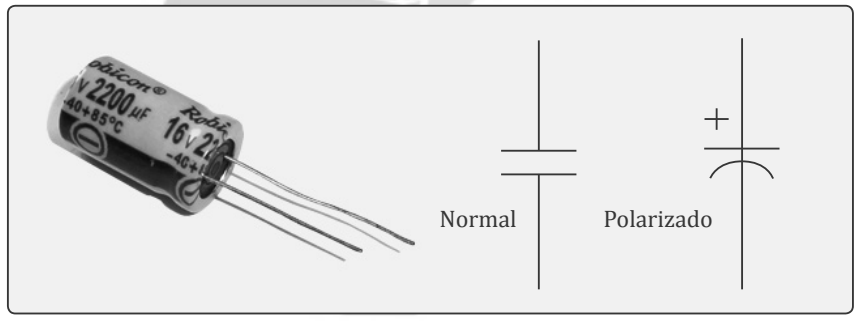


Figura 5.10 El capacitor electrolítico, comúnmente empleado en circuitos. A la derecha se muestra su símbolo

g. Interruptor: Es también conocido como *switch* y los hay de diferentes tipos, pesos y tamaños. Normalmente, se clasifican de acuerdo a su uso. La función de un interruptor es permitir o restringir el paso de la corriente de un punto a otro. Este elemento maneja valores discretos, pues maneja solo dos posibles estados o niveles de tensión existentes en la electrónica digital: HIGH y LOW.

Los tipos de interruptor (*mini_switches*) más populares empleados en la electrónica e ingeniería de hardware son:

- **Interruptor SPST:** Es el interruptor más simple y el más utilizado, sobre todo en hogares. Los hay de diversos tamaños y proporciones. El *switch* SPST (*single pole single throw*) es, comúnmente, utilizado para hacer demostraciones del funcionamiento de compuertas lógicas (simulación). Existen de igual modo los interruptores DPDT (*dual pole dual throw*).

- **Interruptor de pulsación o *push* botón:** Es un componente electrónico, que al efectuar una pulsación, la lámina conductora interna del elemento se acciona y hace posible el paso de la corriente (mediante el cierre de su mecanismo de flujo). Lo anterior posibilita el encendido de lámparas, motores, *buzzers*, *display*, etc. Actualmente es común el uso de interruptores de *push* de dos y cuatro terminales.

- **Interruptor DIP:** Consta de un arreglo de pequeños interruptores ligados entre sí que constituyen una doble línea de contactos, de allí el nombre DIP (*dual inline package*). Estos son usados para trabajar con circuitos compuestos por varias entradas.
- **Otros interruptores:** Se encuentran el *switch* de palanca, utilizado en fuentes de alimentación, pueden poseer varias posiciones y terminales. Otro tipo de interruptor común es el interruptor deslizable, que posee de dos, tres, cuatro posiciones. Son usados en ciertos cargadores o eliminadores para la selección de voltajes o canales[1].

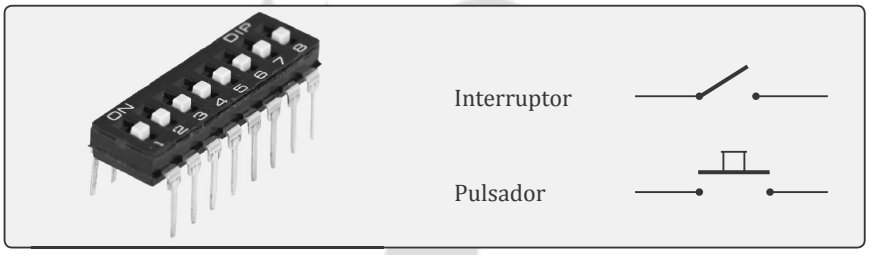


Figura 5.11 El símbolo empleado para representar un interruptor depende del tipo de *switch*. A la derecha se muestran dos símbolos

h. Sensor: Conocido también como captador, se define como un dispositivo que tiene como propósito detectar ciertas magnitudes (generalmente, del entorno ambiental) y transformarlas a pulsos eléctricos que pueden ser cuantificados. Hace un manejo exclusivo de magnitudes físicas llamadas variables de instrumentación como temperatura, intensidad de luz, distancia, gas, aceleración, desplazamiento/movimiento, proximidad, presión, fuerza, humedad, lluvia, etc.

Actualmente, se desarrolla este tipo de elementos para su interacción con placas de hardware libre como Arduino.

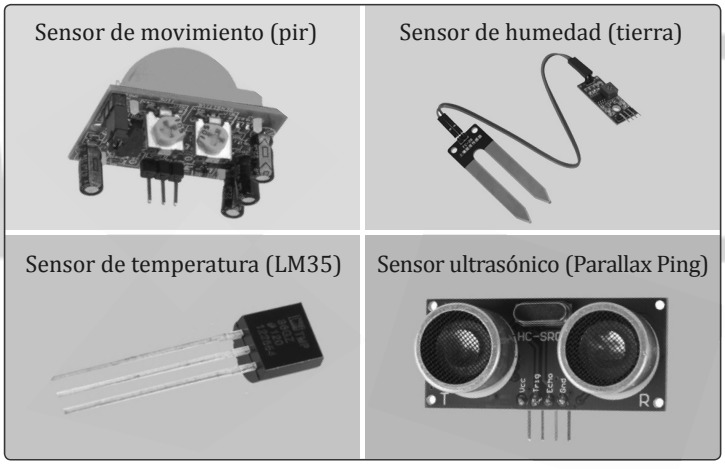


Figura 5.12 Imagen representativa de algunos sensores utilizados en la plataforma Arduino

[1] Para ahondar más en el tema de componentes electrónicos, véase Thomas, F (2013), «Dispositivos electrónicos».

i. Visualizador: Es un componente electrónico conocido como *display*. Permite la visualización de datos (por ejemplo cualquier número o carácter ASCII). Es, comúnmente, utilizado en la electrónica digital y en sistemas informáticos.

Existen diferentes tipos de *display* en el mercado, desde los que son capaces de representar dígitos decimales (0 - 9) hasta mostrar una interfaz gráfica completa e interactiva (pantalla-táctil). Los más representativos son:

- ***Display* digital:** Es un elemento que permite la visualización (o trazado mediante líneas) de dígitos del 0 al 9 de manera individual (así como algunos caracteres de acuerdo a la disposición y número de segmentos). Existen displays digitales compuestos por 7, 14 y 16 segmentos, formados por una línea dispuesta tanto en forma vertical como horizontal. Estas líneas no son más que ledes que mantienen una conexión interna, tal que pueden representar un número ocho. Algunos ejemplos de aplicación son relojes digitales, contadores, toma turnos, letreros, etc.

  Otro tipo de display digital son las matrices, que permiten la visualización de datos mediante ledes dispuestos en un arreglo de filas y columnas. Con ello se pueden realizar anuncios luminosos, semáforos, secuencias de led, etc.

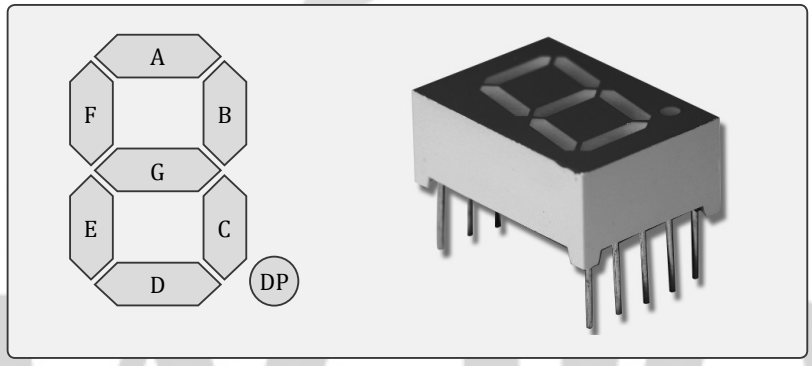


Figura 5.13 Apariencia de un display digital y la distribución de sus siete segmentos

- ***Display* LCD:** Son pequeños módulos segmentados en filas y columnas capaces de mostrar un conjunto de caracteres alfanuméricos. Los tipos más comunes son *display* LCD de 16 × 2 y 20 × 4. A menudo, se utilizan para mostrar información como la temperatura de un entorno, precio de algún producto, avisos o recordatorios, etc.
- **GLCD:** Son pantallas LCD gráficas, su apariencia es muy similar a la anterior, solo que este tipo de *display* es capaz de mostrar gráficos. Las hay de diferentes tamaños y resoluciones, aunque una de las más comunes es la de 128 × 64. Son usadas para mostrar gráficas, figuras básicas monocromáticas y texto.

- **OLED:** Son módulos más complejos, capaces de mostrar cualquier dato o imagen. Algunos estándares manejan una resolución de 128 x 54 y expresan su dimensión en pulgadas (por ejemplo 0.98”). Se usan, comúnmente, para mostrar la hora, la fecha, secuencias de caracteres, permite también la visualización de íconos y figuras a color.
- **Pantalla táctil:** Conocida como LCD touch. Este tipo de *display* es usada en conjunto con plataformas de hardware para su programación o configuración (al igual que las antes citadas). Por lo tanto, permite la manipulación táctil de su panel. Contiene una GUI (*graphic user interface*) y permite la visualización de un sistema operativo completo. Los hay de diferentes resoluciones, 2.8”, 3.2”, 3.5” (480 x 320), 4.3”, 7” (800 x 480), y tecnología, TFT, PiTFT, son ejemplos de ello.

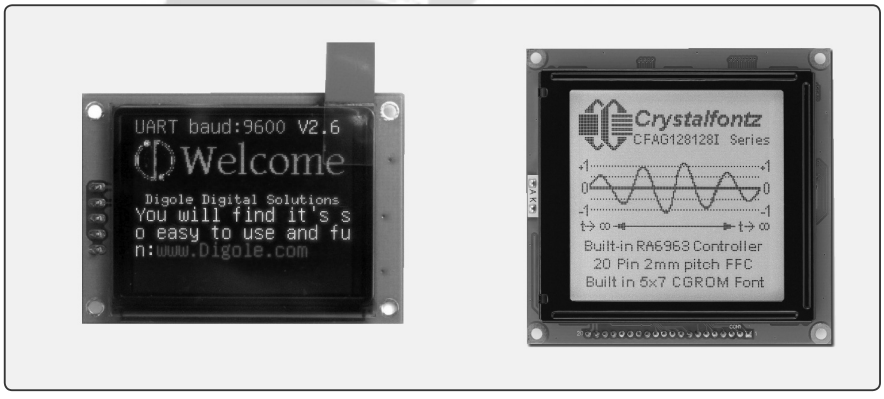


Figura 5.14 Display OLED y display GLCD, respectivamente

5.4 MONTAJES SOBRE PROTOBOARD

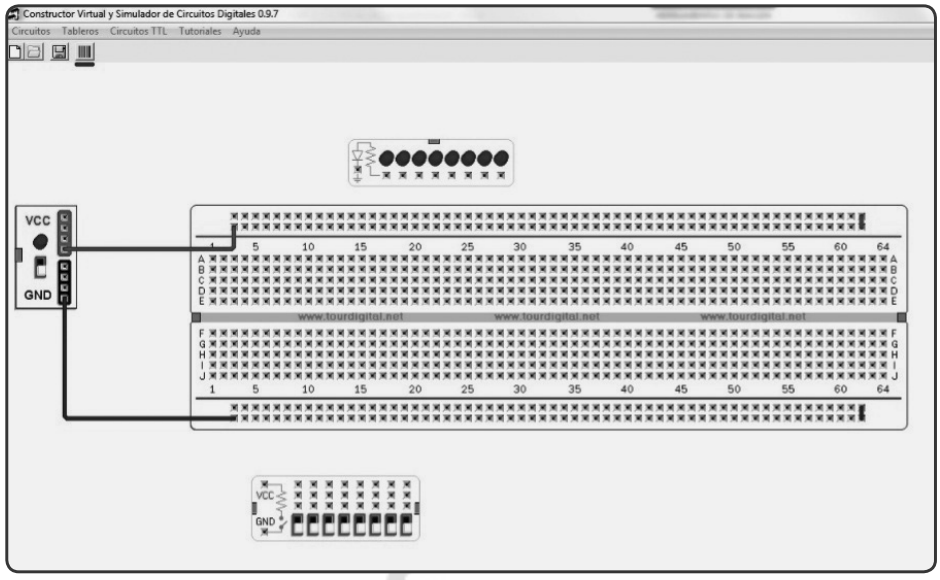
Antes de efectuar cualquier montaje sobre un protoboard físico, es necesario contar con un diagrama topológico funcional, ya que facilita su implementación. Normalmente, se puede recurrir al uso de herramientas virtuales para el diseño de un diagrama topológico. A continuación, se ilustra con un ejemplo el uso de la herramienta virtual que simula esta placa de montaje.

Ejemplo de uso de protoboard

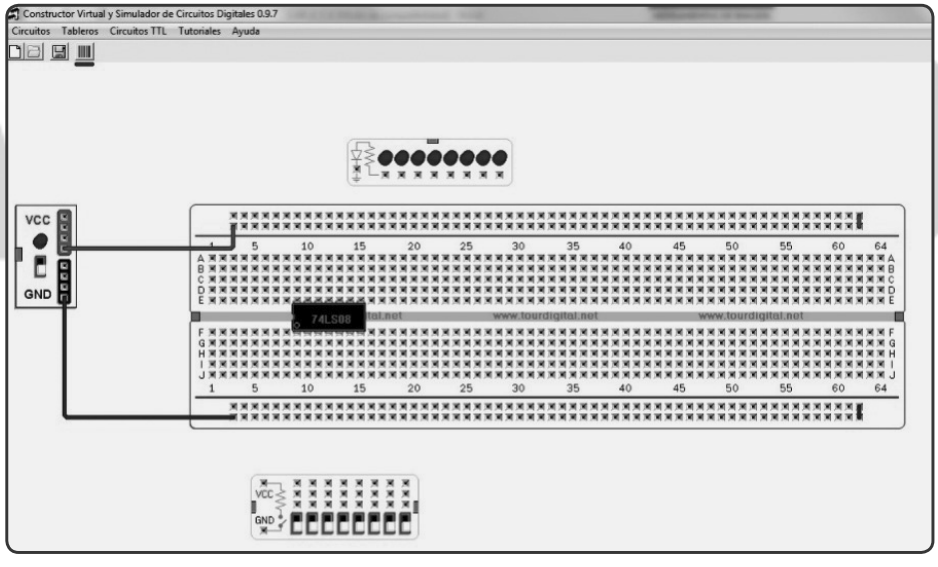
Montar un CI 7408 sobre protoboard para comprobar, mediante una tabla de verdad, el resultado de la función F. Recordando que dicho circuito se asocia a la implementación de una compuerta AND de 2 entradas (x4).

1. Una vez abierto el simulador debe insertarse el material necesario sobre el área de trabajo. Inicie colocando un tablero (equivalente a protoboard), un módulo de led y un módulo de interruptores. Después debe polarizar el protoboard para su funcionamiento. Para ello, es necesario arrastrar un cable de la fuente de alimentación hacia la línea asignada para VCC y GND, respectivamente.

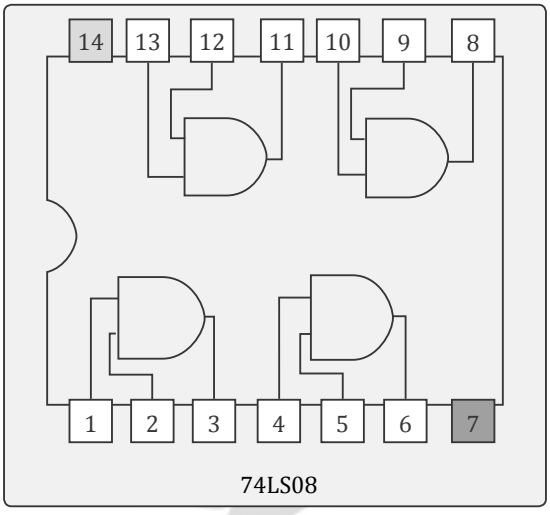
Nota: Para la inserción de cables y componentes debe explorar la barra de menú de la utilería de software.



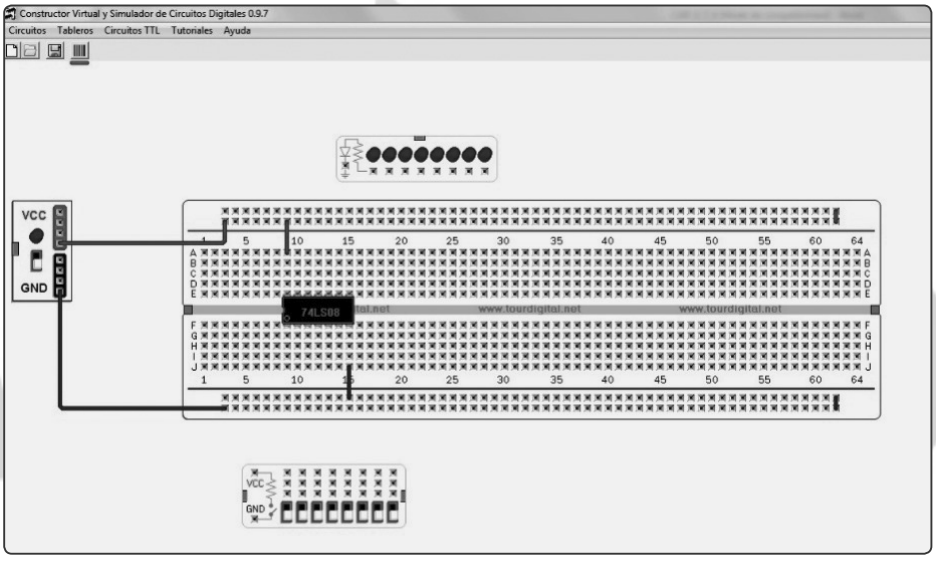
2. Posteriormente, agregue el C.I correspondiente (7408 o compuerta AND de 2 entradas). Para eso, oprima clic sobre el menú circuitos **ttl>puertas básicas>and** y agregue el chip antes mencionado. Finalmente, arrástrelo hacia el área central del protoboard, sobre la línea que divide las columnas centrales.



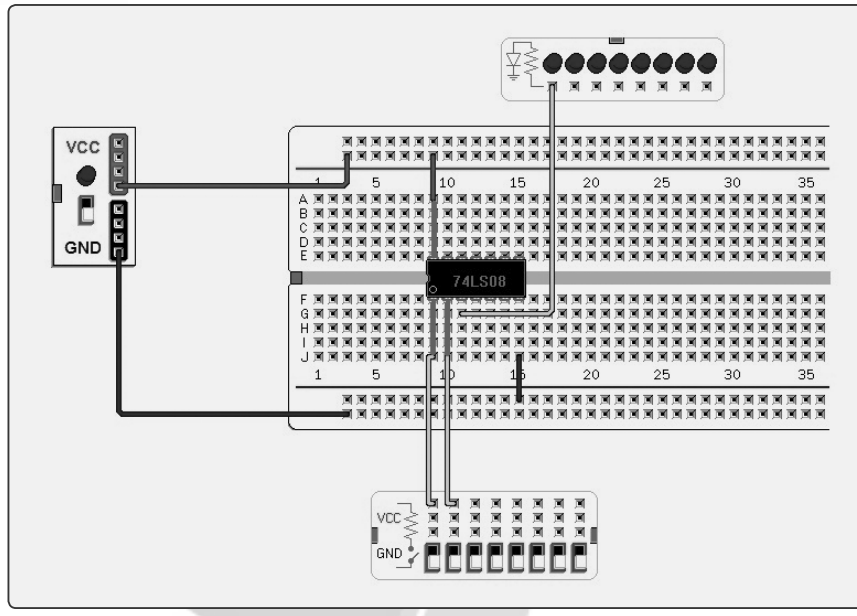
3. Polarice el circuito integrado. Para ello, seleccione el pin destinado a la conexión de VCC y GND correspondiente (tal y como se muestra en el esquema de conexión interna del CI). Por lo tanto, en este caso se debe asociar la línea GND al pin número 7 del circuito y VCC al pin número 14.



74LS08



4. Posteriormente, debe conectar las entradas y salidas correspondientes a la compuerta que desea cablearse. Para conseguir esto, debe analizar los esquemas de conexión de cada componente u hoja de datos (*datasheet*). En este caso se analiza el esquema interno del circuito 7408. Como se sabe las entradas se conectan a los interruptores y la salida a un led.



Para comprobar su funcionamiento, debe tener a la mano la tabla de verdad correspondiente.

Cuando se desea trabajar con *protoboard* físico, es necesario tener en consideración la forma en que debe conectarse cada componente. Por lo general, cada uno de ellos posee, por lo menos, dos terminales que corresponden al polo positivo y negativo **(que deben estar conectados en el sentido correcto)** y están asociados a la línea VCC y GND correspondiente.

Hay que tener presente que la disposición de los orificios existentes en un *protoboard* sirve de guía al usuario para conectar de manera correcta. Si se analiza la conexión **del paso a paso anterior** sobre el constructor virtual, es evidente la falta de resistencias tanto a las entradas como a las salidas. Esto supone un problema al momento de cablear físicamente. A continuación, se muestra un circuito montado sobre *protoboard* físico.

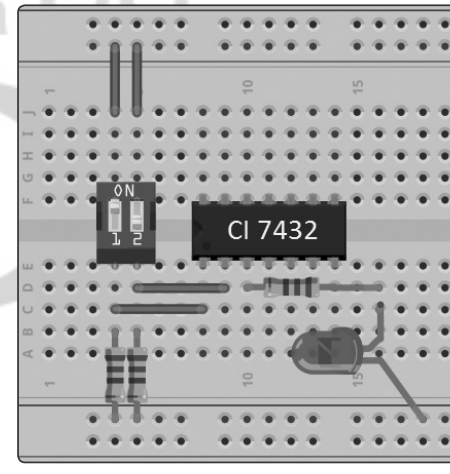


Figura 5.15 Esquema de conexión de una compuerta AND sobre protoboard

Si se analiza la figura anterior, se puede apreciar la forma en que está conectado el led y su respectiva resistencia. Por lo general, existen dos formas de conectar un led a una resistencia. En la siguiente figura se muestra la disposición de ledes sobre protoboard:

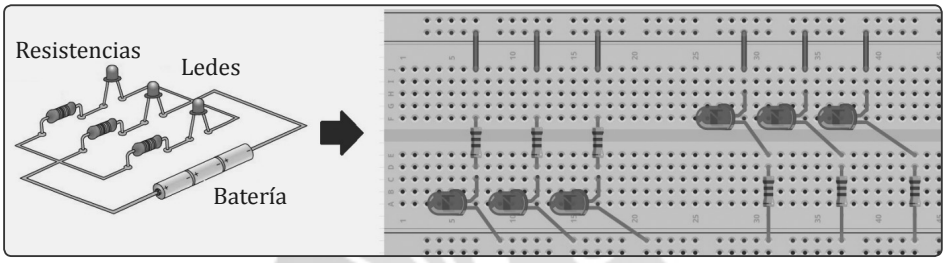


Figura. 5.16 Disposición de ledes sobre un protoboard

La disposición de los ledes en la figura anterior muestra una conexión en paralelo. Cualquiera de ambos casos va a depender del circuito que se desarrolle, aun así, es recomendable seguir el esquema debido para evitar cortos o, simplemente, que los componentes dejen de funcionar de manera correcta.

ACTIVIDAD 1

1. Efectúe el montaje del esquema anterior, tanto de modo virtual (a través del constructor o *WinBreadboard*) como físico.
2. Agregue un *pushbotton* para conseguir el apagado y encendido de cada led (demostración física).
3. Conecte al arreglo un transistor para mantener el control de flujo de la corriente que enciende los led.
4. Argumente conclusiones.

5.5 SHIELDS DE CONEXIÓN

El término *shield* comienza a tener fuerza desde que surgen las plataformas de hardware libre, como Arduino, Raspberry Pi, Galileo de Intel, etc. Con el paso del tiempo era cada vez más evidente que las funcionalidades de estas placas (de propósito específico) fueran limitadas, lo que condujo a pensar en la posibilidad del diseño de módulos periféricos para aumentar sus funcionalidades.

Una *shield* es una placa o módulo (muy independiente de los sensores) que, generalmente, se agrega de manera adicional a cualquier circuito electrónico y que tiene la finalidad de expandir sus funciones. En el mercado existe una amplia gama de este tipo de productos, sobre todo para plataformas de interconexión de hardware como las antes mencionadas.

En algunas ocasiones, por sí misma, una *shield* funciona como nexo que permite el enlace con otras plataformas, circuitos electrónicos, dispositivos, e incluso con equipos portátiles o móviles como laptops, smartphone y tabletas electrónicas, o muchas veces con pantallas, televisores, electrodomésticos, etc. A continuación, se mencionan y describen algunos **proyectos/clasificados** desarrollados en Arduino que precisan el manejo de *shields*:

a. Proyectos de robótica: Un ejemplo son los carros control remoto. En este tipo de proyectos es muy común el uso tanto de sensores como de *shields*. El más habitual en estos casos es el módulo inalámbrico (bluetooth, wifi, XBee, WiShield), el cual permite el control mediante un dispositivo portátil. Los carros control remoto o robots diferenciales pueden llegar a concebir el uso de palancas (*Joystick*), las cuales se encuentran disponibles para la mayoría de plataformas de hardware libre.

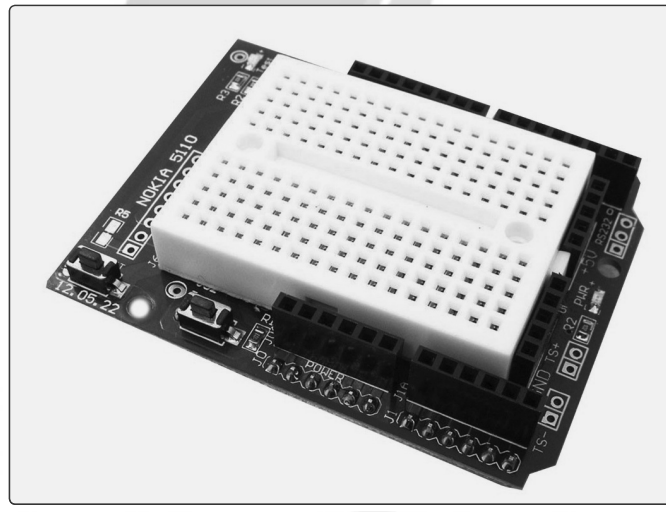


Figura 5.17 La shield de prototipado permite al usuario la conexión de su propio Arduino

b. Proyectos de domótica: El ejemplo más común son las casas automatizadas. En proyectos de domótica o automatización es común el uso de módulos de red (muchas veces inalámbrico), sobre todo si se desea la manipulación de eventos desde una computadora, servidor o panel táctil. En este tipo de proyectos se encuentran muy presentes los sistemas de seguridad de acceso (biométricos, monitoreo) y el uso de sensores (presencia, movimiento, temperatura, etc.), los cuales aprovechan el uso de módulos de interconexión de redes, e incluso existen shields receptoras de sensores.

El encendido de focos, paneles, electrodomésticos y, generalmente, cualquier dispositivo del hogar, edificio u oficina requiere de la intervención de módulos con relevadores (o relés) y el uso de dispositivos móviles. Los paneles de acceso requieren a veces de elementos como una LCD *Keypad* para mostrar o escribir información, ya que contiene algunos pulsadores que permiten la navegación por la pantalla.

c. Proyectos de seguridad: El ejemplo más representativo es una caja fuerte. Como se ha mencionado, los sistemas de seguridad a menudo requieren del uso de tarjetas adicionales o periféricos. Un par de *shields* comúnmente usadas en este tipo de proyectos son un módulo LCD Keypad y un Servo Shield (es una placa controladora de distintos tipos de motores). Esta última suele utilizarse para conectar un servomotor que permita abrir o cerrar la caja. Para visualizar la información, se puede utilizar shield LCD Touch. En otro tipo de proyectos de seguridad suelen utilizarse *shields* para almacenar información sobre tarjetas SD.

d. Proyectos de reproducción de audio y video: Se aconseja el uso de módulos como la MP3 Shield. Consiste en una placa con pulsadores que emulan o representan los botones de reproducción (PREV, NEXT, PLAY) y volumen (VOL+, VOL-). Para visualizar imágenes desde una pantalla tradicional, usualmente, se puede utilizar una shield Video Game o similar.

e. Proyectos de recolección de datos: El chat es un claro ejemplo de proyectos que se encuentran muy involucrados con el manejo del software, aunque se sabe que el hardware es vital, sobre todo si se desea almacenar cierto tipo de información (conversaciones por ejemplo). Para esto se utiliza *shields* SD (tarjeta de almacenamiento). Este tipo de proyecto puede hacer uso adicional de *shields* GPS. En caso de prever una comunicación inalámbrica, se pueden utilizar distintas *shield* que proporcionen la conexión. De no ser así basta con la conexión del propio Arduino.

Algunos otros ejemplos de shields son *GPRS Shield*, *Colors Shield*, *GPS Shield Kit*, *SD Card Shield*, *Ethernet Shield*, *Servo Shields*, *Liquidware TouchShiel* (pantalla táctil), etc.

Una *shield* incluso puede ser diseñada por los propios usuarios. Para esto se requiere de conocimientos básicos de electrónica y desarrollo de hardware. Estas, por lo general, son desarrolladas con el único fin de satisfacer necesidades muchas veces personales o comunes entre cierta comunidad de usuarios.

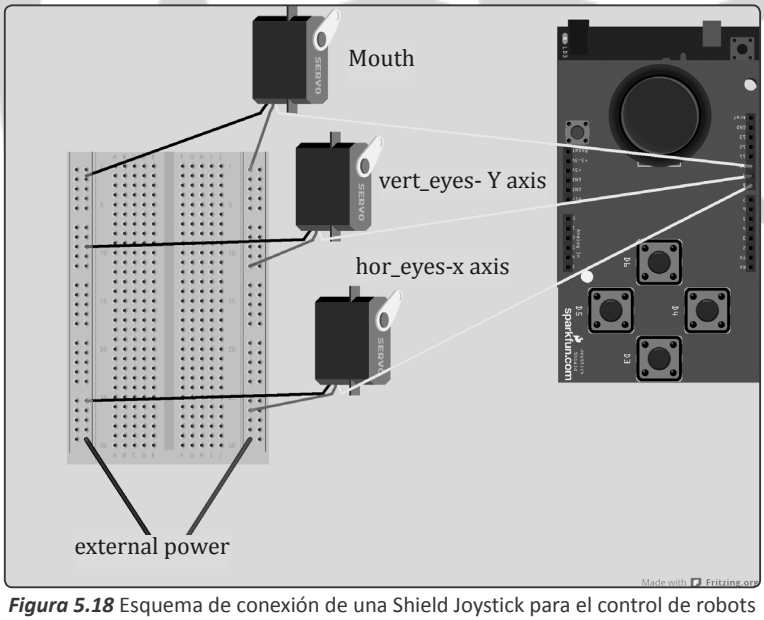
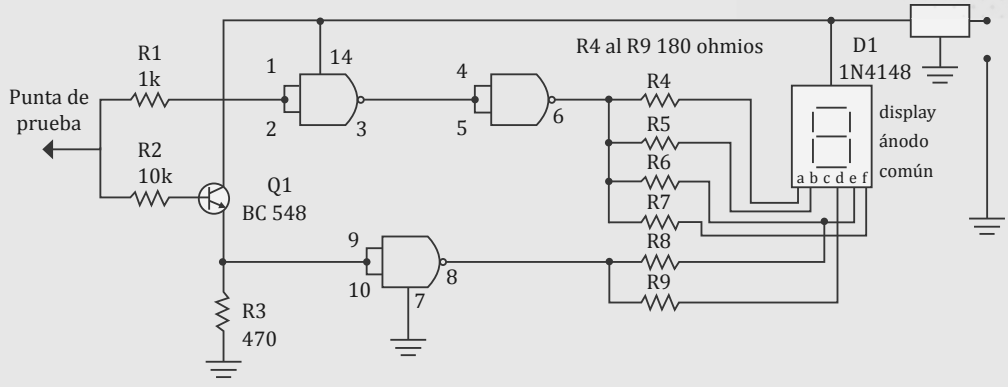


Figura 5.18 Esquema de conexión de una Shield Joystick para el control de robots

ACTIVIDAD 2

Dado el siguiente diagrama (punta lógica TTL con uso de *display* de 7 segmentos) realice lo que se solicita:



a) Realice una lista de los componentes electrónicos que integran el presente circuito.

b) Monte su circuito sobre el constructor y simulador de circuito digitales antes mencionado.

c) Compruebe el funcionamiento del montaje anterior (inciso b) con el uso de LiveWire.

d) Monte sobre protoboard físico dicho circuito.

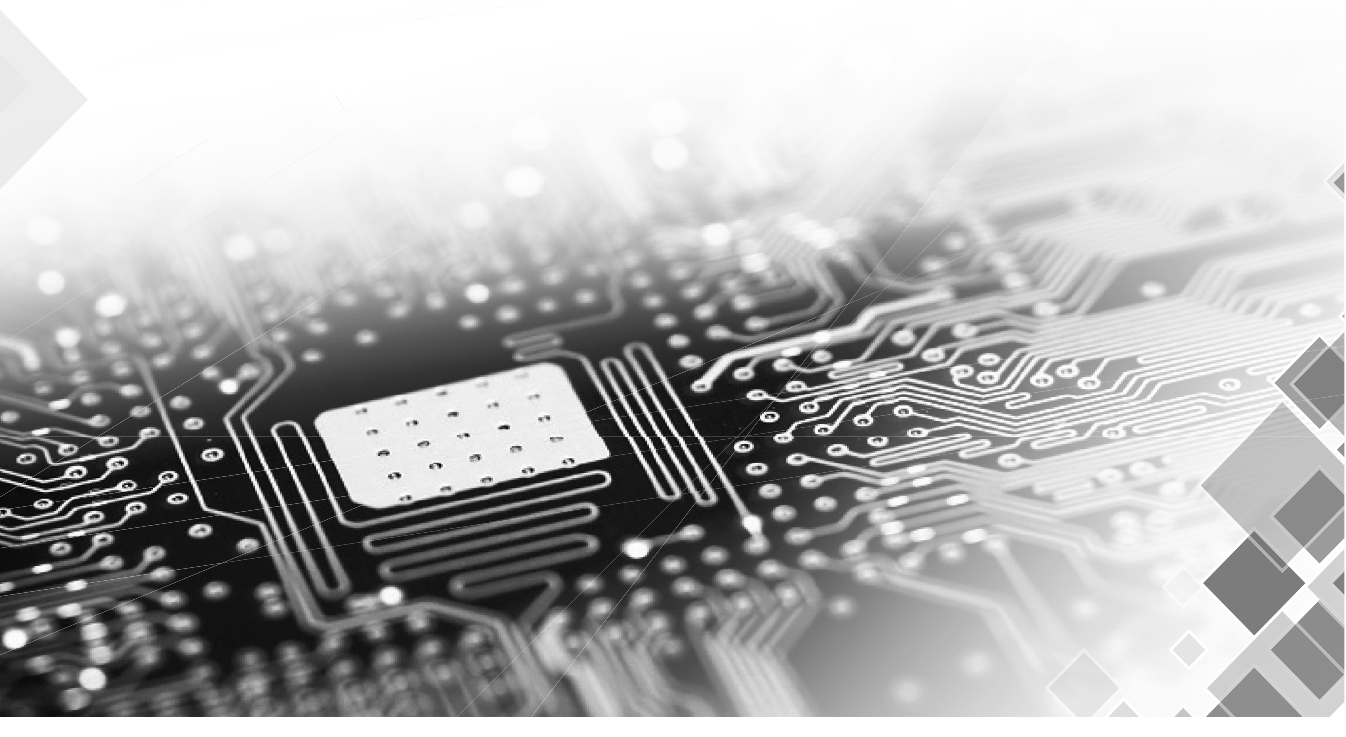
e) Planifique el diseño de una shield que permita la conexión de un display de siete segmentos.

En este capítulo se ha revisado, de manera básica, los componentes electrónicos más utilizados para comenzar a hacer pruebas de funcionamiento de un circuito. Se ha desarrollado el contenido sobre el uso del protoboard, tanto de manera virtual como física. Finalmente, se ha hecho un estudio sobre algunos proyectos desarrollados en la plataforma Arduino y los tipos de shield que utilizan. En el próximo capítulo se atenderá un contenido exclusivo sobre sistemas digitales.

EDITORIAL
MACRO®

Capítulo 6

Sistemas digitales



EDITORIAL
MACRO®

Se entiende por sistema digital a un conjunto de arreglos (generalmente de dispositivos o componentes electrónicos) que tienen como fin la generación y tratamiento de las señales digitales. Un ejemplo es la PC. Existe una gran variedad de sistemas digitales en el campo, aunque la mayoría son bastante abstractos. En conjunto, muchos de ellos brindan soluciones integrales que se hayan presentes tanto en módulos de hardware como dispositivos de distinta índole (electrodomésticos, informáticos, de comunicación, etc.).

Los sistemas digitales por lo general trabajan con los dos niveles de tensión propios de la electrónica digital (HIGH y LOW), además en su formato primitivo le permiten al usuario la construcción de dispositivos, circuitos o proyectos funcionales. Contemplan el uso de dígitos para representar sus salidas.

Figura 6.1 Un contador electrónico o reloj es un claro ejemplo de sistema digital

6.1 EL SISTEMA BCD

El sistema BCD (*binary coded decimal* o decimal codificado en binario) es un sistema de computación, el cual consiste en una simple conversión de un valor expresado en base decimal a su equivalente en binario (4 bits o *nibble*). Esto es posible a través de un arreglo especial de codificación de señales. Existe el código BCD natural, BCD Aiken y BCD Exceso 3.

Durante el manejo y aplicación de un sistema BCD, debe considerarse la distribución de peso, el cual consiste en el conjunto de números que determinan las posiciones de lectura del valor binario resultante, de derecha a izquierda.

- Para el código BCD natural, se emplea la siguiente distribución: 8 4 2 1.
- Para el código BCD Aiken, se emplea la siguiente distribución: 2 4 2 1.
- Para el código exceso 3, se emplea la siguiente distribución: (8 4 2 1) + 3.

La representación del código BCD original estándar (llamado también BCD natural) es la siguiente:

Tabla 6.1 *Disposición de bits para un sistema BCD natural*

Decimal	Sistema BCD natural			
Peso	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

La representación del código BCD Aiken es la siguiente:

Tabla 6.2 *Disposición de bits para un sistema BCD Aiken*

Decimal	Sistema BCD Aiken			
Peso	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

Es importante observar que la disposición de la posición 0 a 4 forma parte de un grupo. Mientras que, de la 5 a la 9 forma parte de un segundo grupo. Para esta última agrupación se deben convertir los 0 lógicos en 1 y viceversa. Con esto se obtiene una simetría en los valores con el primer grupo.

Tabla 6.3 *Disposición de bits para un sistema BCD exceso 3*

Decimal	Sistema BCD natural				Exceso 3			
Peso	8	4	2	1	+ 3			
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Ejemplo de uso del sistema BCD

1. Convierta la siguiente expresión decimal a código BCD natural: 780292d.

Realizando la conversión con base a lo visto en el capítulo dos de este libro en el tema de sistemas numéricos, se obtiene:

Posición del valor decimal	Valor decimal	Sistema BCD (código)			
		8	4	2	1
6	7	0	1	1	1
5	8	1	0	0	0
4	0	0	0	0	0
3	2	0	0	1	0
2	9	1	0	0	1
1	2	0	0	1	0

Queda de la siguiente forma:

780292d = 0111 1000 0000 0010 1001 0010_BCD

Al momento de la conversión deberán considerarse los cuatro bits sin excepción (respetando, de igual modo, la distribución del peso, que en este caso se lee como: 8 4 2 1). Posteriormente, deben acomodarse los valores tal y como lo muestra cada dígito.

2. Convierta la misma expresión decimal a código BCD Aiken: 780292d.

Para realizar la conversión se utiliza la siguiente distribución de peso: 2 4 2 1 (sustituyendo los valores de la tabla anterior).

Queda de la siguiente forma:

780292d = 1101 1110 0000 0010 1111 0010_BCDAiken

3. Convierta la misma expresión decimal a código BCD exceso 3: 780292d.

Para realizar la conversión se utiliza la siguiente distribución de peso: 8 4 2 1 +3 (sustituyendo los valores de la tabla anterior).

Queda de la siguiente forma:

780292d = 1010 1011 0011 0101 1100 0101_BCD exceso 3

En caso de emplear una conversión o codificación inversa (de binario a decimal), se realiza lo siguiente:

Ejemplo de codificación por sistema BCD de binario a decimal

1. 1010b → Decimal por código BCD natural

a. Se verifica el peso. En este caso: 8 4 2 1.

Peso	8	4	2	1
Valor binario	1	0	1	0

b. Se sustituye cada valor binario por su valor decimal equivalente (se considera la suma de valores decimales, cuyo bit sea igual a 1 lógico).

Peso	8	4	2	1
Valor binario	1	0	1	0
Obtención de valores en 1	8	-	2	-

c. La obtención de la suma de acuerdo a la distribución del peso es = 10d.

2. 1010b → Decimal por código BCD Aiken

a. Se verifica el peso. En este caso: 2 4 2 1.

Peso	2	4	2	1
Valor binario	1	0	1	0

b. Se sustituye cada valor binario por su valor decimal equivalente (se considera la suma de valores decimales, cuyo bit sea igual a 1 lógico).

Peso	2	4	2	1
Valor binario	1	0	1	0
Obtención de valores en 1	2	-	2	-

c. La obtención de la suma de acuerdo a la distribución del peso es = 4d.

3. 1010b → Decimal por código BCD exceso 3

a. Se verifica el peso. En este caso: 8 4 2 1 + 3

Peso	8	4	2	1
Valor binario	1	0	1	0

b. Se sustituye cada valor binario por su valor decimal equivalente (se considera la suma de valores decimales, cuyo bit sea igual a 1 lógico).

Peso	8	4	2	1
Valor binario	1	0	1	0
Obtención de valores en 1	8	-	2	-

c. La obtención de la suma de acuerdo a la distribución del peso es 10d + 3 a cada uno de los bits. Obteniendo finalmente:

Peso	8	4	2	1
Valor binario	1	0	1	0
Exceso 3 para 1	0	1	0	0
Exceso 3 para 0	0	0	1	1

d. Resultado por exceso 3: 1 + 3 = 4 y 0 + 3 = 3

ACTIVIDAD 1

1. Convierta mediante sistema BCD (natural, Aiken y exceso 3) las siguientes expresiones:
   - **a)** 98377d: _____
   - **b)** 8872d: _____
   - **c)** 9677210d: _____

2. Convierta de binario a decimal utilizando el sistema BCD (natural, Aiken y exceso 3):
   - **a)** 1001d: _____
   - **b)** 1111d: _____
   - **c)** 10110d: _____

6.2 USO DE DISPLAYS

A través de los *displays* es posible la visualización de datos expresados en cualquier base numérica. Por ello, es uno de los elementos indispensables en el mundo de la electrónica y la ingeniería de hardware.

Como ya se ha revisado, existen diferentes tipos de *displays* (desde los *displays* de 7 segmentos, hasta las pantallas táctiles de última generación), los cuales son empleados según la necesidad de cada usuario o exigencias del proyecto.

La conexión interna de los *displays* de 7 segmentos se encuentra dispuesta en dos formas, ánodo común y cátodo común.

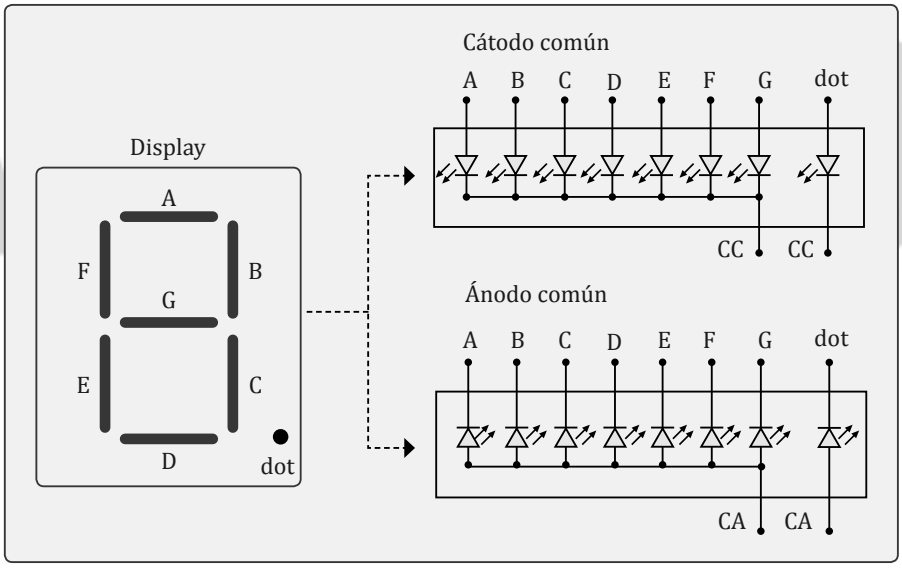


Figura 6.2 Conexión interna de un display de 7 segmentos

Los *displays* de 7 segmentos son muy utilizados en electrónica, sobre todo en sistemas de codificación (en el que se hallan inmersas diferentes bases o sistemas como código BCD), que hacen uso a su vez de temporizadores, *flip flops* (los cuales se verán más adelante en este mismo capítulo).

La forma más usual de visualizar datos, construido algún circuito con contadores, relojes, codificadores o decodificadores (los cuales se apreciarán en el siguiente tema de este capítulo), es a través del *display*. A continuación, se muestran algunos formatos:

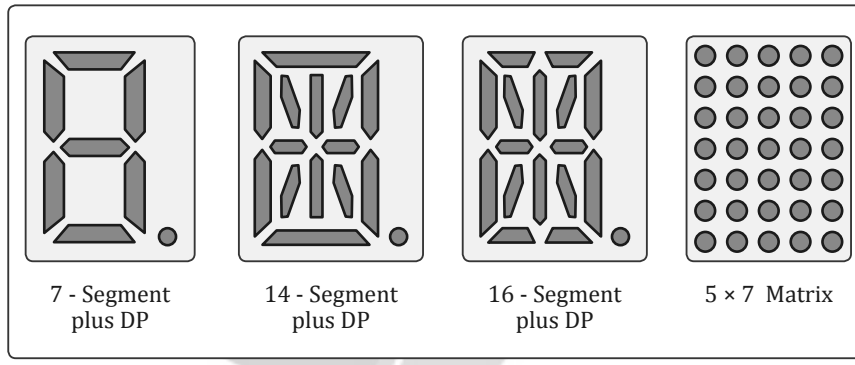


Figura 6.3 Gama de displays existentes en el mercado

Los *displays* de este tipo suelen venir en distintas presentaciones, una de las más actuales es en conjunto de cuatro muy usados en plataformas de hardware libre para programar relojes, cronómetros, secuencias numéricas, etc.

6.3 CIRCUITOS COMBINACIONALES Y SECUENCIALES

Dentro del gran universo de los sistemas digitales, se cuenta con circuitos combinacionales y circuitos secuenciales. Su diferencia principal radica en el valor de sus salidas dependientes de sus entradas. A continuación, se describe de manera general la diferencia de este tipo de circuitos digitales.

6.3.1 Circuitos combinacionales

Se denomina sistema combinacional a todo aquel sistema digital en el que sus salidas dependen directamente del valor de sus entradas en un momento dado. Lo anterior sin la intervención de estados anteriores tanto de las entradas como de las salidas. Las funciones AND, OR, NAND y XOR son por lo general representadas por una tabla de la verdad. Por lo tanto, carecen de memoria y de retroalimentación. Sin embargo en los circuitos secuenciales, los valores de las salidas, no dependen exclusivamente de los valores de las entradas en ese momento, sino también dependen del estado anterior o estado interno. Por ello se añade un bloque de almacenamiento (memoria) que guarda ese valor interno que a su vez se suma al circuito combinacional para obtener una salida.

A. Codificadores y decodificadores

a. Codificadores: Los codificadores son circuitos combinacionales que sirven para hacer una conversión de información expresada de forma numérica o alfanumérica a un sistema binario. Entre los codificadores más empleados se encuentran el CI74147 (10 líneas a 4) y el CI 74148 (8 líneas a 3 más GS y E0), ambos codificadores de prioridad.

Para comprender mejor el funcionamiento de un codificador, se propone un ejemplo de aplicación:

Ejemplo de aplicación de un codificador de 10 a 4 líneas

Se desea expresar en valor binario los valores de un teclado numérico (0-9 dígitos).

Para efectuar tal tarea, debe plantearse una tabla de verdad, la cual contenga 10 dígitos (que fungen como entradas), para lo cual se establecen 4 salidas (que expresan 4 bits, suficientes para expresar una cantidad no mayor a 15).

Entradas (pulsadores)										Salidas (ledes)			
0	1	2	3	4	5	6	7	8	9	F4	F3	F2	F1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

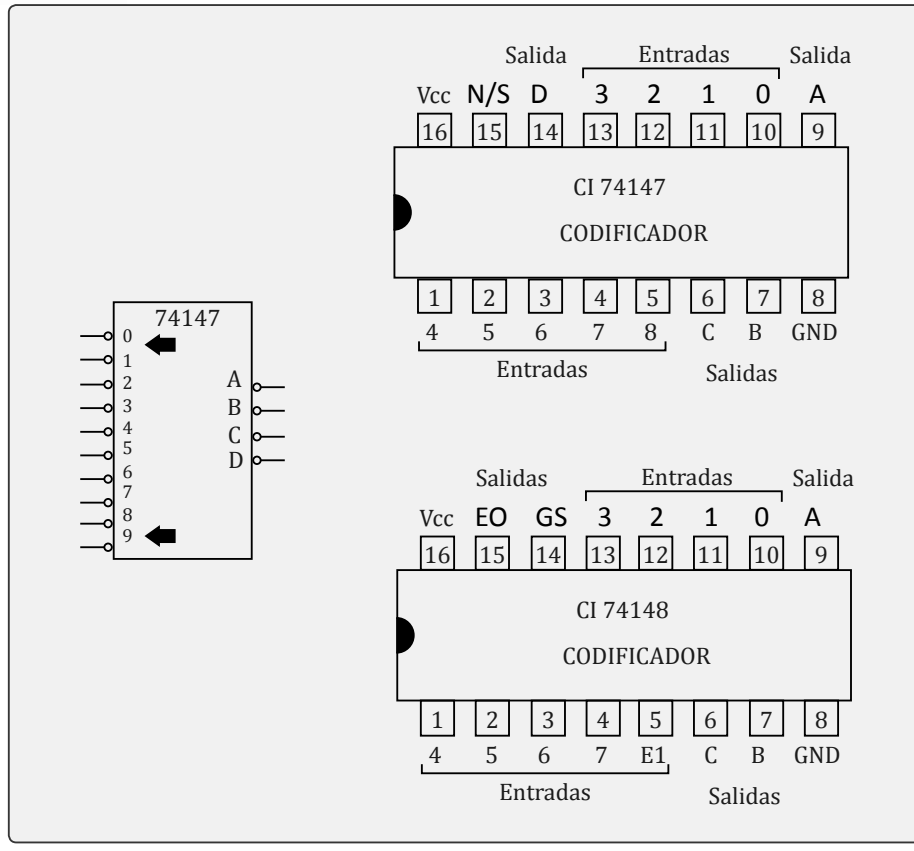


Figura 6.4 Esquema de conexión del CI 74147 y 74148

Como se puede apreciar, cuando una de las entradas está en 1 lógico, quiere decir que el pulsador correspondiente está activado. Por lo tanto, las salidas reflejarán el número de ledes que se encenderán de acuerdo a los valores implícitos en el sistema binario.

b. Decodificadores: Los decodificadores o descodificadores son circuitos combinacionales que sirven para hacer la conversión entre distintas bases (binario, decimal, BCD). Actualmente, existen diversos códigos utilizados en la electrónica digital, y entre ellos podemos destacar BCD-8421 código Gray, código de Johnson, el código ASCII, entre otros.

A menudo se desarrollan proyectos de decodificación utilizando circuitos integrados como el CD4511 (BIN a DEC), el CI7442 (BCD a DEC), CI7447/CI74247 (BCD a 7 segmentos), etc. En estos casos, es importante recordar que las salidas también pueden ser llevadas a *display* de 7 segmentos.

Los decodificadores pueden ser manejados desde plataformas como Arduino. Más adelante se hará la explicación correspondiente a este tipo de implementaciones.

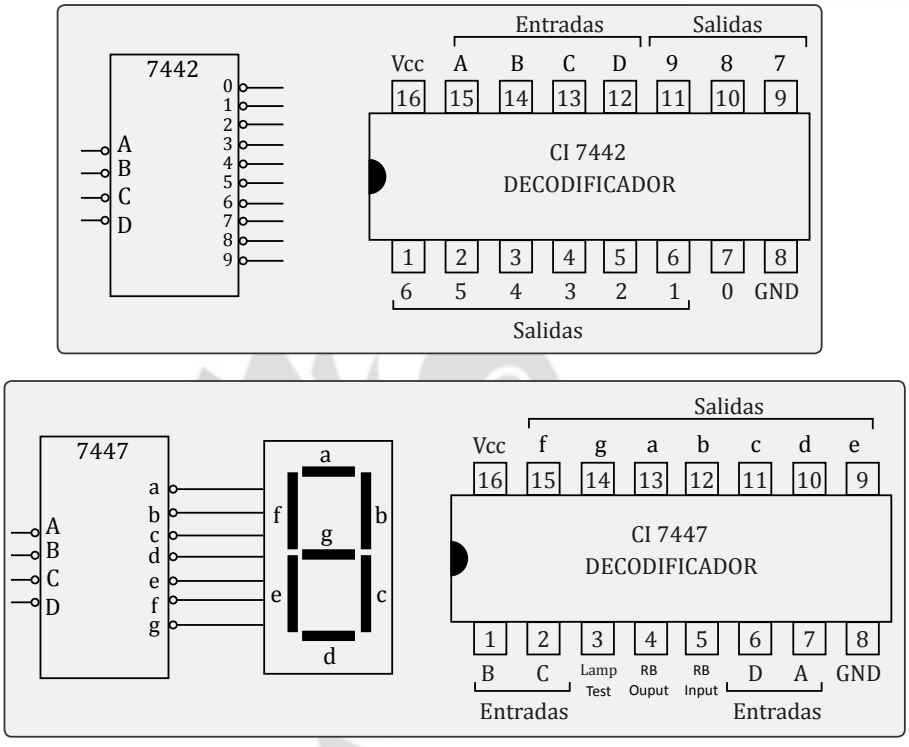


Figura 6.5 Esquema de conexión del circuito decodificador CI 7442 y 7447

Existe una gran gama y tipos de decodificadores en el mercado, entre los que destacan son decodificador 1 a 2, 1 a 4, 1 a 8. Un ejemplo de este último es el decodificador 74LS138.

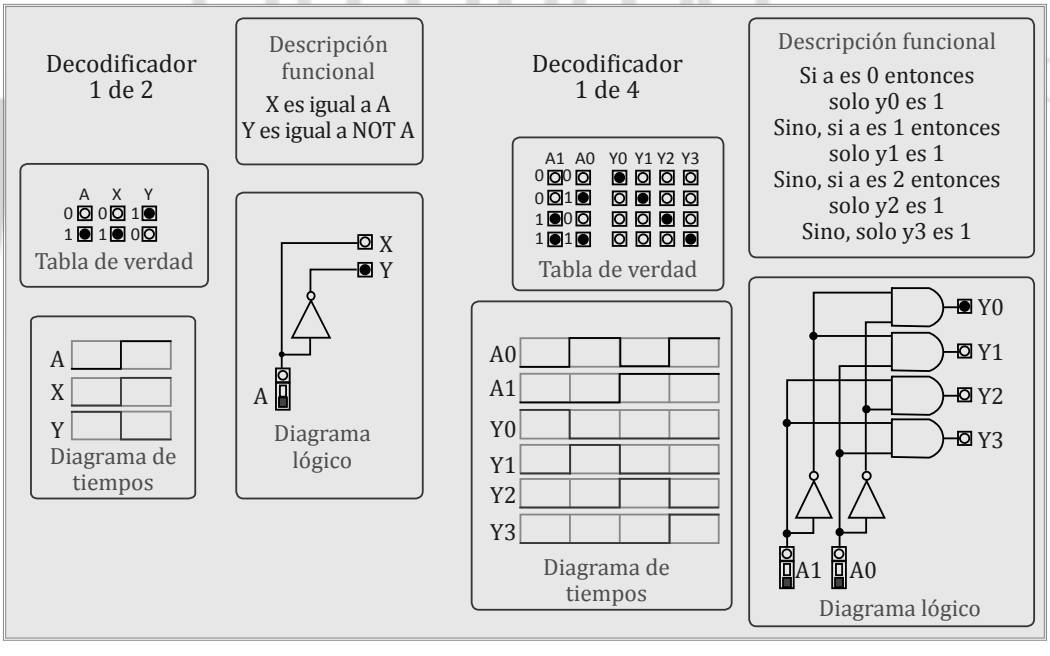


Figura 6.6 En esta figura se ilustran algunos tipos de decodificador

Práctica de laboratorio n.º 6

Decodificadores y codificadores

Objetivo de la práctica: Crear un codificador y un decodificador para hacer conversiones de base 2 a base 10 y viceversa (haciendo uso de display de 7 segmentos y ledes según corresponda).

Fundamento teórico:
Un decodificador se encarga de identificar, reconocer o bien detectar un código específico. Lo opuesto a este proceso de decodificación se denomina codificación y es realizado por un circuito lógico que se conoce como codificador.
Un codificador tiene varias líneas de entrada, solo una de las cuales se activa en un momento dado y produce un código de salida de N bits, según sea la entrada que se active. Estos nos permiten efectuar conversiones numéricas de diferentes bases.

Procedimiento:
1. Para la creación de un codificador de decimal a binario.

Materiales:

Será planteado por el estudiante:

_____
_____
_____
_____
_____
_____

2. Para la creación de un decodificador de binario a decimal.

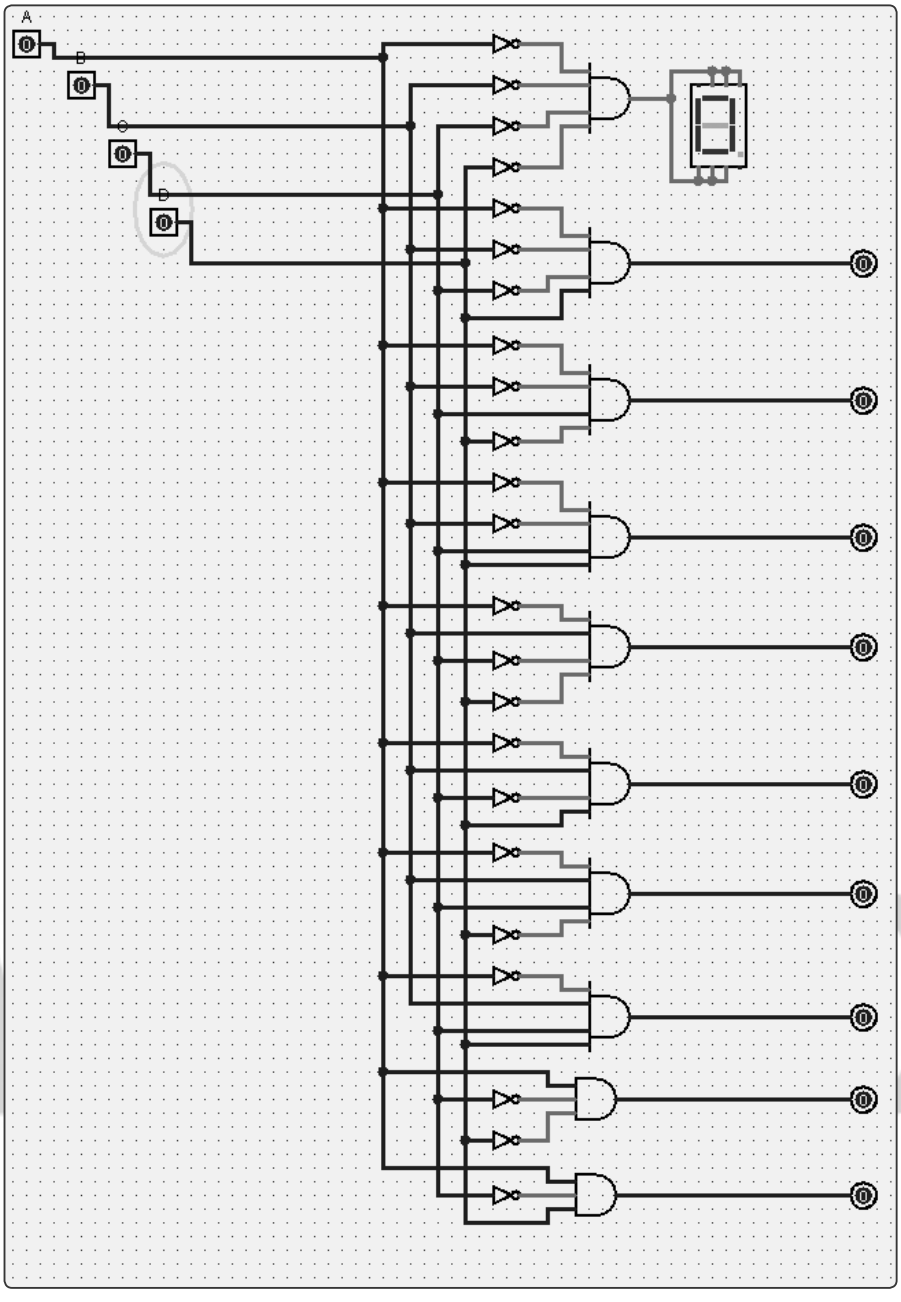
Materiales:

Será planteado por el estudiante:

_____
_____
_____
_____
_____
_____

Cuestionario inicial

1. Dada la siguiente figura (conversor de binario a decimal), conteste:



a) Diseñe una tabla de verdad para el presente decodificador.

b) Obtenga una expresión lógica válida (para 0, 1, 2, 3… 9).

c) Haga la implementación del mismo circuito con uso de un display de 7 segmentos.

2. Cree un circuito que permita realizar la conversión de un sistema decimal a binario de 4 bits (codificador).
   a) ¿Cuál es el nombre del CI que utilizará para esta actividad?
   b) Dibuje el diagrama lógico de este circuito.

Para el ejercicio 1 y 2:

a) Monte el circuito topológico sobre el constructor y simulador de circuitos digitales.
b) Monte el circuito en protoboard físico.

Cuestionario final

1. Mencione por lo menos tres diferencias existentes entre un codificador y un decodificador.
2. ¿En qué consiste un decodificador y codificador de paridad?
3. Realice un esquema de conexión de un CI 7447 a un *display* de 7 segmentos.
4. ¿Qué utilidad tiene el CI 74138?

B. Multiplexores y demultiplexores

a. Multiplexores: Un multiplexor o MUX se define como un circuito compuesto por varias entradas y una única salida de datos. Su funcionamiento consiste en seleccionar solo una de las entradas de datos para permitir su transmisión hacia su única salida. Se encuentra integrado por N valores de selección, los cuales excitan dicha salida. Algunos multiplexores utilizados en la electrónica digital son el CI 74151 (8 x 1), 74153 (4 x 1), 74157/8 (2 x 1).

Ejemplo de multiplexor

En el diagrama lógico de un multiplexor de cuatro entradas y una salida (MUX de 4 X 1) está constituido por cuatro compuertas AND, dos NOT y una OR.

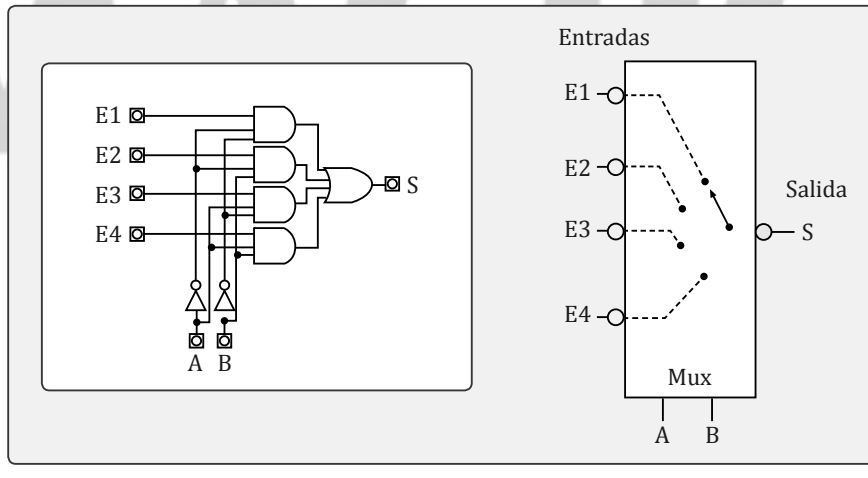


Figura 6.7 Diagrama lógico y representación de un MUX 4 X 1

La tabla de verdad y la expresión booleana de este circuito se arma con base a los selectores (A, B) y las entradas que en este caso son cuatro (E1-E4). Tal y como se aprecia a continuación:

A	B	S
0	0	E1
0	1	E2
1	0	E3
1	1	E4

$$F = (A' B' E1) + (A' B E2) + (A B' E3) + (A B E4)$$

El circuito integrado a utilizar, dependerá del número de entradas del circuito MUX. Para la construcción de su tabla y expresión booleana se sigue el mismo principio.

El proceso de multiplexación, habitualmente, se aplica en varios campos del conocimiento como en telecomunicaciones, en informática, en electrónica, robótica, etc. Los principales objetivos de la multiplexación son:

- Compartir la capacidad de transmisión de datos sobre un mismo enlace para aumentar la eficiencia.
- Minimizar la cantidad de líneas físicas requeridas y maximizar el uso del ancho de banda de los medios.

Nota: Actualmente, existen CI que incorporan la funcionalidad tanto de un decodificador como de un multiplexor (es el caso del CI 74LS138).

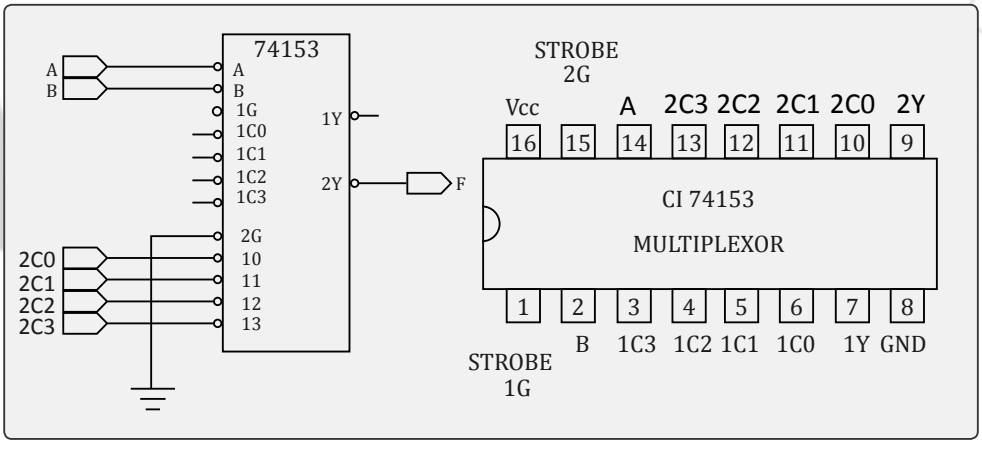


Figura 6.8 Esquema de conexión del MUX 4 X 1

b. Demultiplexores: También llamado DEMUX, se trata de un circuito combinacional integrado por una sola entrada de datos y N entradas de control. Posee además N canales de salida. Representa el complemento de un multiplexor (proceso inverso). Algunos ejemplos de demultiplexor son: CI 74HC138, 74HCT138, 74HC238, 74HCT238.

Ejemplo del diagrama lógico de un demultiplexor de una entrada y cuatro salidas (DEMUX de 1 X 4)

Está constituido por cuatro compuertas AND y dos NOT.

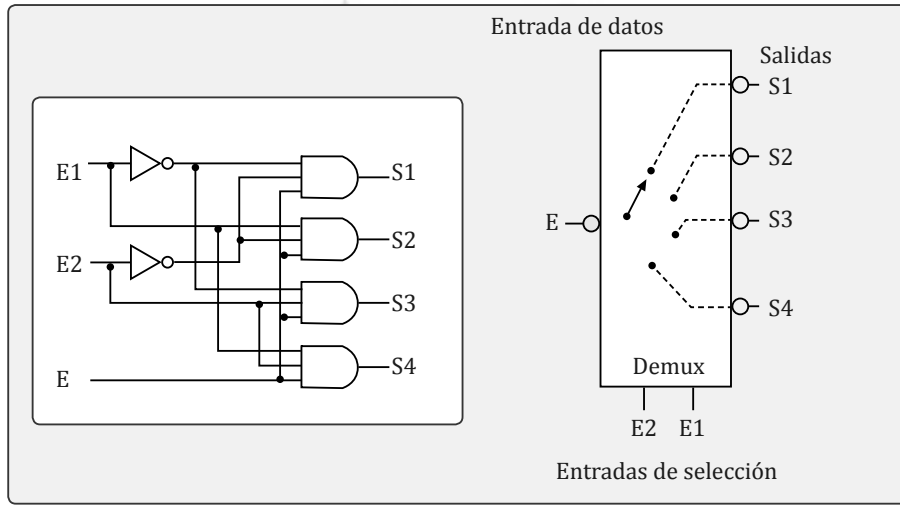


Figura 6.9 Diagrama lógico y representación de un DEMUX 1 X 4

La tabla de verdad y la expresión booleana de este circuito se arma con base en los selectores (E1, E2) y las entradas que, en este caso, se representa como E. Tal y como se muestra continuación:

ENTRADAS			SALIDAS			
E2	E1	E	S1	S2	S3	S4
0	0	E	E	0	0	0
0	1	E	0	E	0	0
1	0	E	0	0	E	0
1	1	E	0	0	0	E

Se debe tener en cuenta que E puede adoptar un valor lógico de 1 o 0. Lo que ofrece como resultado un conjunto de funciones para cada salida:

S1 = E2’ E1’ E
S2 = E2’ E1 E
S3 = E2 E1’ E
S4 = E2 E1 E

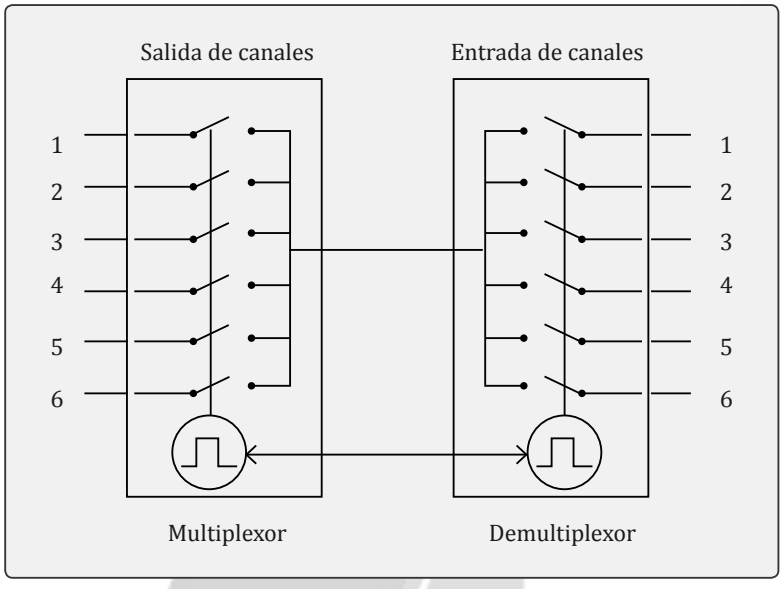


Figura 6.10 Esquema de conexión general entre un MUX y un DEMUX

Nota de interés

Motor de búsqueda de *datasheet*

Un recurso que no puede faltar en la mesa virtual de trabajo, de cualquier usuario dedicado a la electrónica, es un motor de búsqueda de *datasheets* u hoja de datos. En la siguiente dirección puede consultar cientos de hojas de datos de distintos componentes electrónicos http://www.alldatasheet.es/.

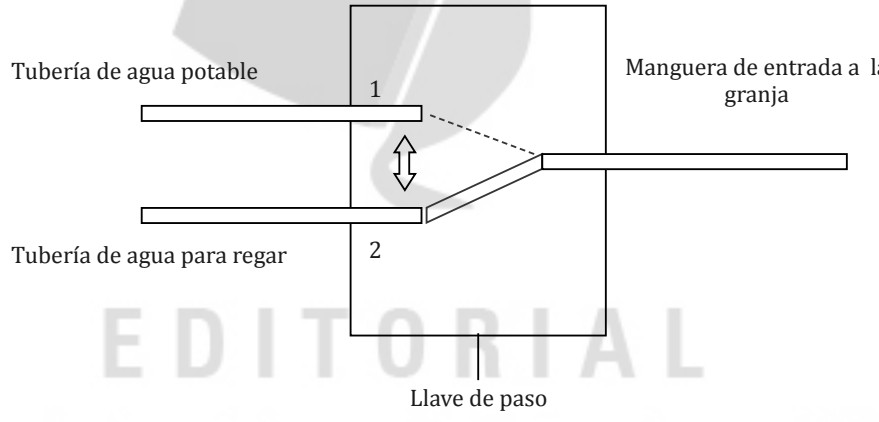
Práctica de laboratorio n.º 7

Multiplexores y demultiplexores

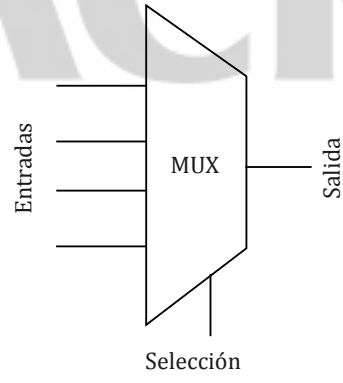
Objetivo de la práctica: Comprobar el funcionamiento de un multiplexor (MUX) y un demultiplexor (DEMUX) mediante montajes físicos y simulación.

Fundamento teórico:

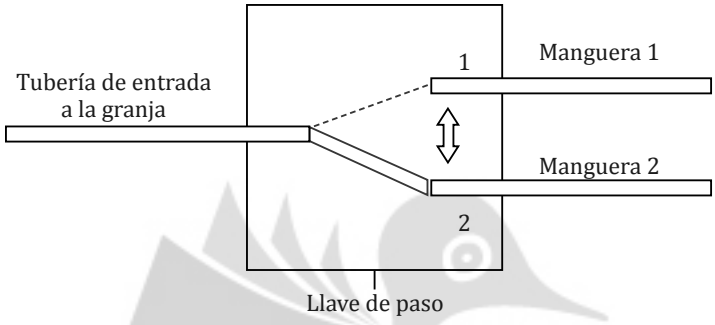
Un **multiplexor** está constituido por varios canales o vías, por donde puede transitar un bit a la vez (se llaman entradas de datos), por selectores y una única salida. Esto significa que se puede seleccionar el dato en la salida del componente que se está utilizando. Un ejemplo de aplicación para entender el funcionamiento de un multiplexor es un selector de agua potable o agua para riego. La cual implica la selección de un solo canal para despachar agua, el cual se encuentra en función de una llave de paso para conseguir la salida.



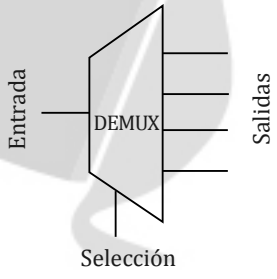
El esquema típico digital de un MUX es:



Un **demultiplexor** se constituye de un único canal de entrada de datos, cuyo bit tiene que salir por una única vía de múltiples salidas. Para comprender mejor esta definición, se propone el uso del ejemplo anterior (proceso inverso complementario):

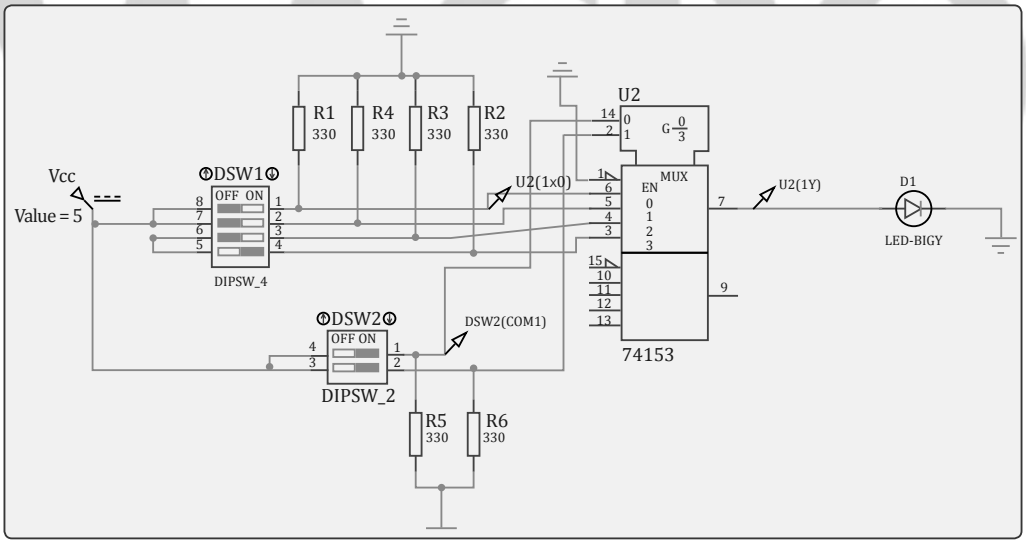


El esquema típico digital de un DEMUX es:

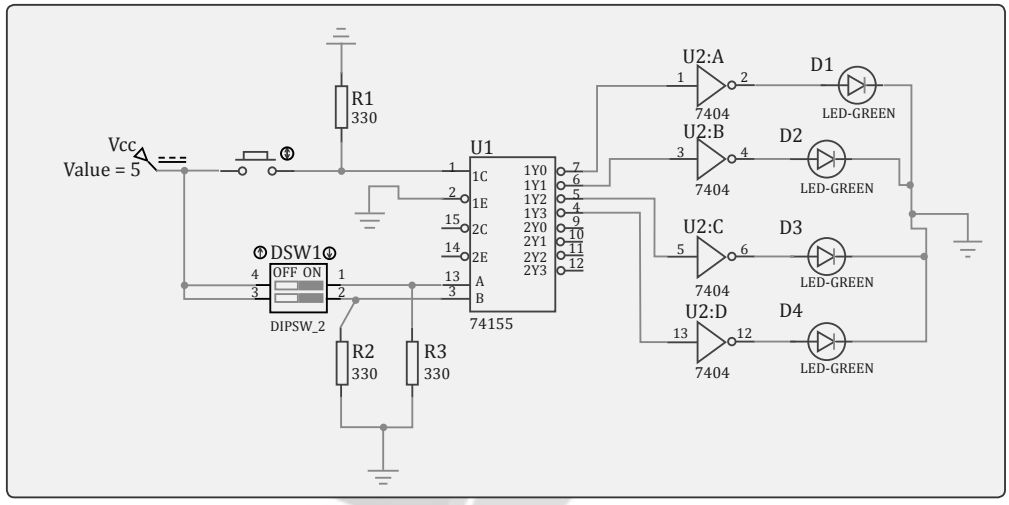


Procedimiento:

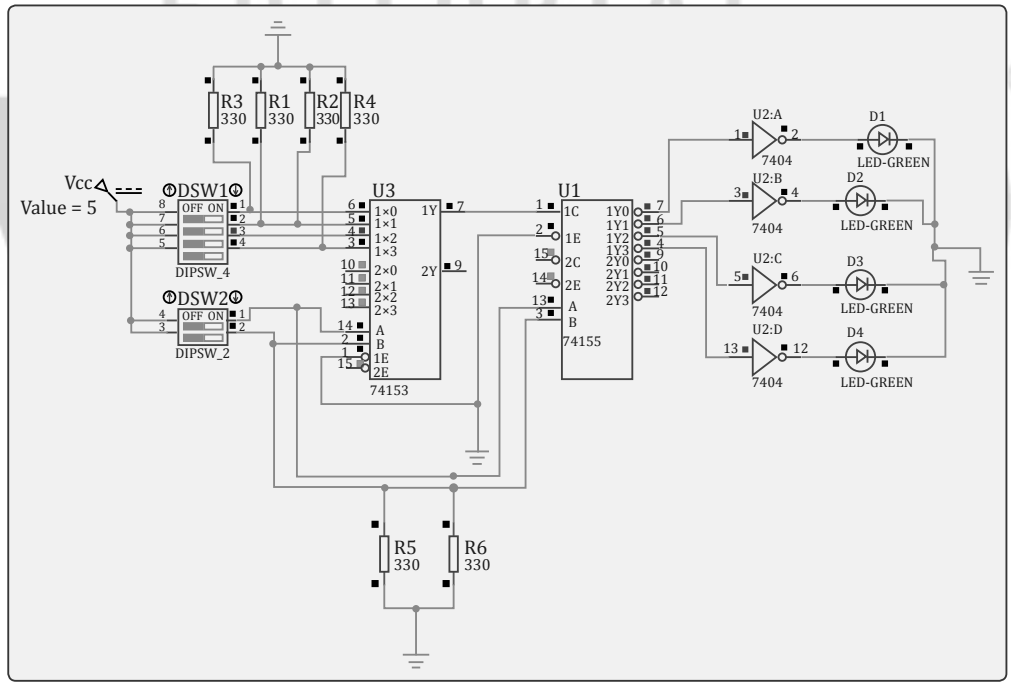
1. Efectúe el montaje del siguiente diagrama con el fin de comprobar su funcionamiento. Realice la tabla de verdad del multiplexor 74LS153.
   - **a)** Se requiere montaje sobre herramienta de simulación.
   - **b)** Se requiere montaje físico.



2. Realice el siguiente diagrama para comprobar el funcionamiento del demultiplexor 74LS155. Realice su tabla de verdad correspondiente.
   a) Se requiere montaje sobre herramienta de simulación.
   b) Se requiere montaje físico.



3. Realice el siguiente montaje:
   a) Determine el material utilizado.
   b) Describa el funcionamiento del circuito.
   c) Realice el montaje en herramienta de simulación.
   d) Realice el montaje de manera física.



C. Comparadores

Un comparador es un circuito combinacional que tiene como fin comparar dos entradas mediante el uso de operadores relacionales.

Ejemplo: Dadas las entradas A y B, respectivamente, comparar según: A<B, A>B O A=B.

Uno de los comparadores de mayor fama es el comparador de 4 bits, el cual se implementa en un CI74LS85. La relación de sus pines es la siguiente:

A>B, A<B, y A=B: entradas de comparación en cascada activas a nivel alto (1).

A>B, A<B, y A=B: salidas de comparación activas a nivel alto (1).

A0, A1, A2, A3: entradas del dato A.

B0, B1, B2, B3: entradas del dato B.

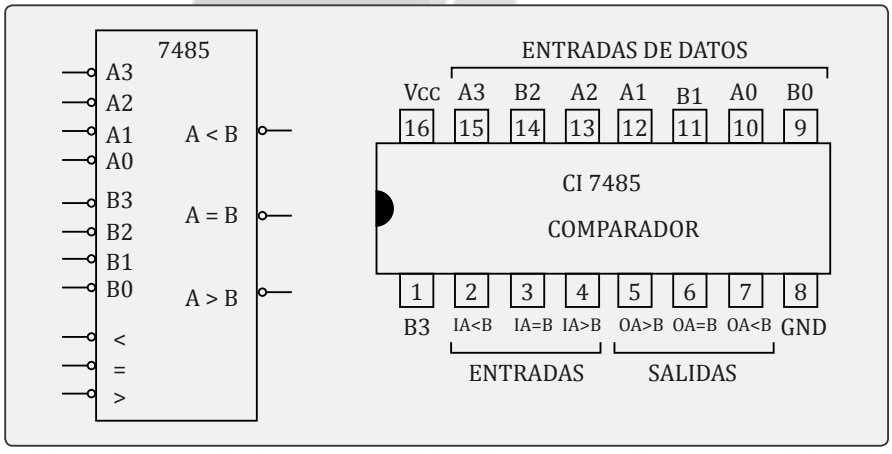


Figura 6.11 Esquema de distribución de pines del CI7485

En el circuito 7485 existen unas entradas en cascada (A>B, A<B y A=B) que sirven para aumentar la capacidad del comparador, es decir, para conectar otro comparador en cascada y comparar datos de 8 bits. El primer comparador compara los 4 bits menos significativos (parte baja del dato) y sus salidas se conectan a las entradas en cascada del comparador superior, que compara los 4 bits más significativos (parte alta del dato). Así, si los datos de la parte alta son iguales, el comparador de la parte baja informa si esta es inferior, superior o igual.

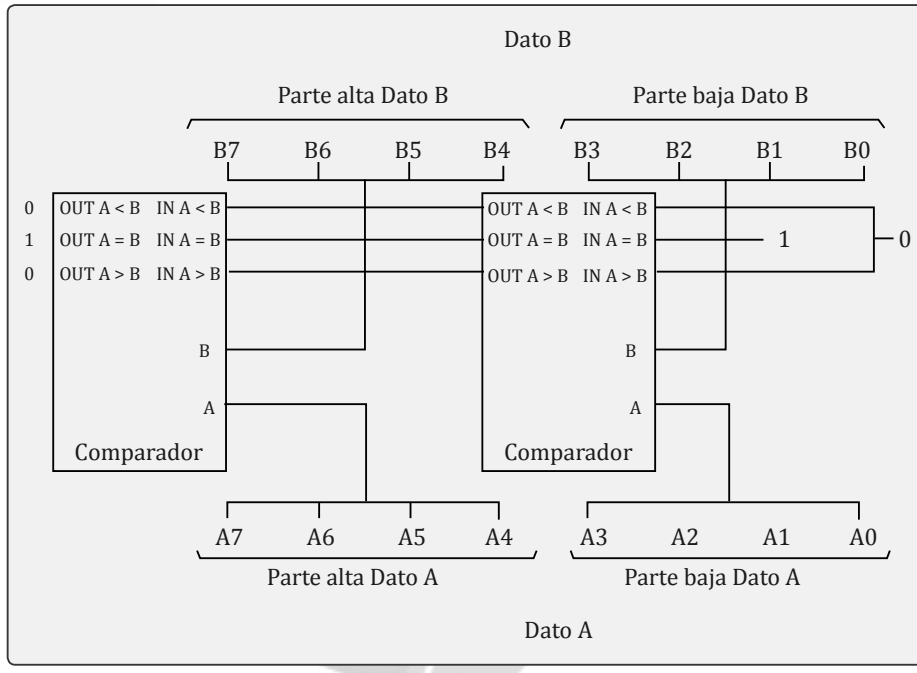


Figura 6.12 Esquema de distribución de pines del CI7485

La tabla de verdad de un comparador de 4 bits es la siguiente:

Datos comparados				Entradas en cascada			Salidas		
A3, B3	A2, B2	A1, B1	A0, B0	A < B	A = B	A > B	A > B	A = B	A < B
A3>B3	X	X	X	X	X	X	1	0	0
A3<B3	X	X	X	X	X	X	0	0	1
A3=B3	A2>B2	X	X	X	X	X	1	0	0
A3=B3	A2<B2	X	X	X	X	X	0	0	1
A3=B3	A2=B2	A1>B1	X	X	X	X	1	0	0
A3=B3	A2=B2	A1<B1	X	X	X	X	0	0	1
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	1	0	0
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	0	0	1
A3=B3	A2=B2	A1=B1	A0=B0	0	1	0	0	1	0

Nota de interés

Operadores relacionales

Existe una gran variedad de operadores relacionales, entre los que destaca el >, < e =. Estos, a veces, son utilizados en el ámbito de la electrónica digital para comparar valores, aunque también se encuentran presentes en el ámbito de la algoritmia y la programación, pues, gracias a ellos es posible no solo hacer comparaciones, sino efectuar estructuras selectivas. Actualmente, son muy utilizados en programación de plataformas de hardware.

D. Sumadores y restadores

El circuito integrado 7483A y el CI 74283 son sumadores binarios por excelencia, permiten el desarrollo de sumas de 4 bits. Existen otros de 2 bits y sumadores semicompletos. El montaje de estos circuitos facilita la obtención de operaciones, mismas que pueden ser visualizadas sobre ledes o, en su defecto, sobre *display*.

Un circuito sumador está conformado por un arreglo de conexiones internas que permite la obtención de resultados.

En la figura siguiente se puede apreciar la forma en la que se encuentra distribuidas una serie de compuertas, de tal manera que, tienen como propósito ofrecer el resultado de una suma binaria, teniendo para esto dos entradas y una salida por bit (sin contar Ci y Co) y, en conjunto, ocho entradas y cuatro salidas por circuito. Ci (input) y Co (output) son acarreos que se originan al momento de realizar la operación. Estos aseguran que la suma represente un valor de 4 bits más uno adicional para almacenar el valor del conteo siguiente.

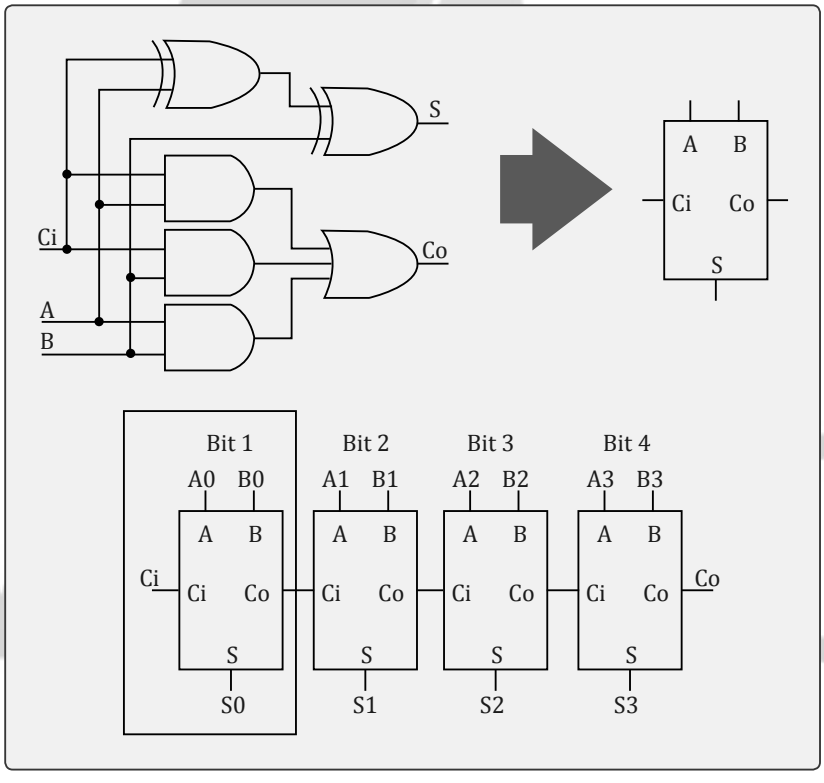


Figura 6.13 Conexión interna de un sumador de 4 bits y su representación

Para comprender tanto la forma de conexión de los pines del CI7483A/283 como su funcionamiento, se recomienda consultar su *datasheet* correspondiente. En esta se pueden ubicar diagramas como los que se muestran en la siguiente figura:

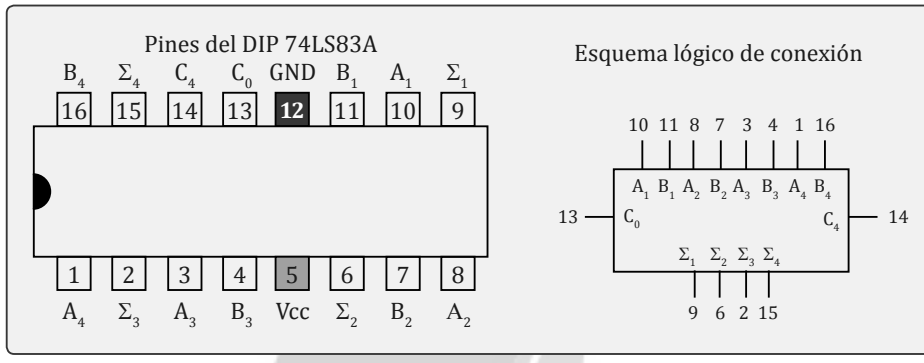


Figura 6.14 Esquemas de conexión de un circuito 7483A

Como se puede apreciar, en el lado izquierdo se tiene el esquema de conexión de las terminales del circuito integrado en cuestión (sobre el que se señala los pines de polarización, entradas, salidas y, Ci y Co correspondientes). En el lado derecho se aprecia la distribución lógica de entradas y salidas del sumador binario de 4 bits. Es importante notar que el número total de terminales de este tipo de circuito es de ocho.

Para la realización de restas binarias se sigue el mismo procedimiento citado en el capítulo dos de este libro. En este caso, el esquema de conexión cambia, aunque se puede desarrollar con el mismo circuito integrado (CI 7483A u 74283). Para esta tarea es importante agregar los componentes necesarios que se encargarán de hacer el cambio de operación. El siguiente esquema representa un circuito sumador-restador.

Circuitos con sumadores

Se sabe que los circuitos que implementan sumadores binarios, por lo general pueden ser convertidores BCD a BCD exceso 3 y sumadores-restadores de 4 bits.

a. **Convertidor BCD - BCD exceso 3:** Una aplicación directa de un sumador de cuatro bits como el CI7483 es un convertidor BCD a BCD exceso 3 que se puede realizar sumando al dato de entrada A, una constante B = 3 (0011b) como se muestra a continuación:

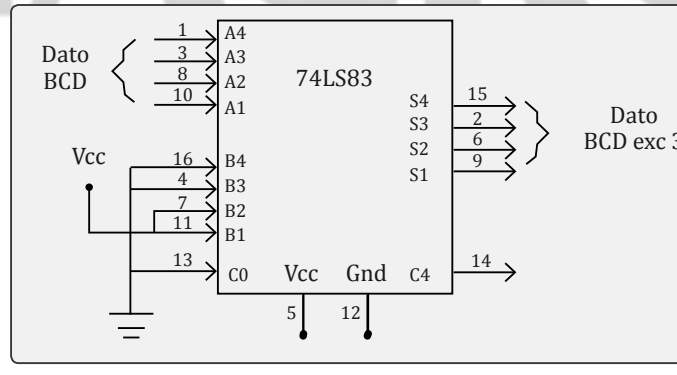


Figura 6.15 Esquema tradicional de un circuito sumado de 4 bits

b. Sumador - sustractor de 4 bits: Para efectuar una resta de A-B usando la suma de A + complemento a dos de B se puede realizar un sumador-restador de cuatro bits como se aprecia en la siguiente figura:

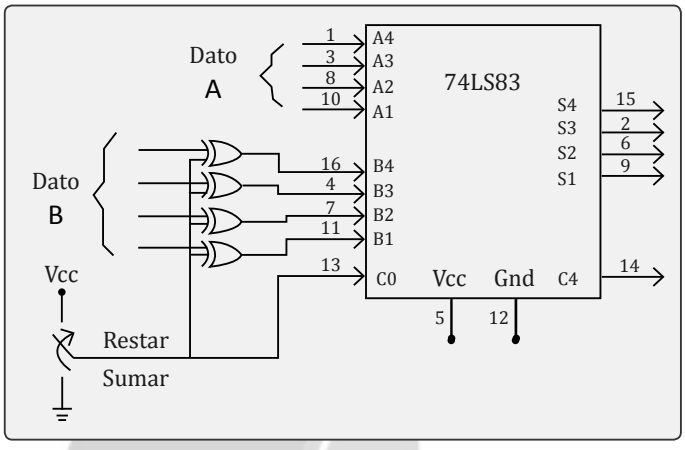


Figura 6.16 Esquema tradicional de un sumador-restador de 4 bits

Como se puede observar existe un bloque de cuatro compuertas XOR, las cuales se encargan de realizar el complemento a uno del dato B cuando el interruptor se encuentra en la posición de resta binaria. Y Co le suma 1 a este complemento, a uno de B para obtener su complemento a 2.

ACTIVIDAD 2

1. Realice un circuito combinacional minimizado con puertas lógicas que implemente un circuito que realice el complemento a dos de un número de cuatro bits.

   Fundamento teórico

   El complemento a dos de un número binario de n dígitos es su diferencia a 2^n. Así el complemento a 2 del número N será $\mathbf{2^n - N}$. Para n = 4 dígitos, será su diferencia a 24 = 16. Por ejemplo, el complemento a dos del número 13 = 1101 es el 16 - 13 = 3 = 0011. El complemento a dos de un número binario se puede obtener también intercambiando el conjunto de 0 lógicos por 1 lógicos y viceversa y sumándole 1 al dígito que ocupa la posición de menor peso. Se recuerda que para convertir un número binario a decimal se multiplica cada dígito por 2 elevado a su posición y luego se suman. El dígito más a la derecha tiene la posición 0 y se sigue contando hacia la izquierda. Por ejemplo: 1001 = 9.
2. Implemente, con el uso de decodificadores de cuatro entradas, un circuito detector de paridad impar (número de **unos** impar) de un número digital de 5 entradas.
   - **a)** Construya tabla de verdad (de 5 entradas y una salida). La salida es 1 lógico cuando exista un número impar de unos por combinación (fila).
   - **b)** Minimice por mapas de Karnaugh.

3. Diseñe un sistema combinacional capaz de realizar la multiplicación de dos números binarios, de un dígito cada uno. El sistema tendrá una señal de acarreo de entrada, y una señal de acarreo de salida.
   a) Minimice el sistema combinacional.
   b) Describa el método de generalización del problema para números de un número mayor de dígitos binarios.
4. Implemente un circuito de 5 variables de entradas con decodificadores de 4 entradas.

   F = f(A, B, C, D, E) = m1 + m2 + m15 + m16 + m17 + m28 + m29 + m30 + m31

5. Realice con multiplexores de cuatro canales y puertas la función lógica siguiente:

   F = f (A, B, C, D) = m0 + m4 + m5 + m6 + m7 + m9 + m13 + m15

6.3.2 Circuitos secuenciales

La aplicación más común de un circuito secuencial se encuentra presente en contadores y memorias digitales.

Actualmente, existe una clasificación de estos sistemas en cuanto a su comportamiento, estos pueden ser asíncronos y síncronos.

a. **Sistemas asíncronos:** A menudo actúan de forma continua en función del tiempo. Cualquier cambio de las entradas provoca cambios en las variables internas sin esperar la intervención de un reloj. Son sistemas más complejos de diseñar.

b. **Sistemas síncronos:** Consiste en un sistema en el que las variables internas no cambian hasta que llega un pulso del reloj (CLK o clock) que no es más que un tren de pulsos periódico. Estos actúan en forma discreta en función del tiempo.

   Se debe tener presente que el cambio de las variables internas se puede producir de dos maneras en un sistema secuencial síncrono. Cuando las señales se validan por un estado lógico (nivel alto o bajo) de la señal de reloj, se dice que son activadas por nivel. Cuando se produce las validaciones de las señales, en el momento que la señal de reloj cambia de estado, se dice que son activadas por flanco de subida (cambio de nivel bajo a alto) y flanco de bajada (cambio de nivel alto a bajo).

A. *Flip flops*

El elemento de memoria básico de los circuitos secuenciales es el biestable, también llamado *flip flop* (multivibrador/oscilador). Se constituye de dos estados estables de funcionamiento y se encarga de almacenar un estado 0 o un estado 1. Se encuentran disponibles tanto de forma asíncrona como síncrona.

Existen diferentes tipos de biestables. Cada uno de ellos dentro de un grupo de clasificación:

Asíncronos: *Flip flop* RS

Síncronos: *Flip flop* RS, J-K, T y D

a. Biestable RS asíncrono (activo al nivel alto): Entre los biestables más populares se encuentra el *flip flop* RS, el cual se conforma (en su forma más básica) de dos entradas: S (*set*) y R (*reset*), y dos salidas complementarias llamadas Q y Q'. Posee, además, una entrada CLK (reloj) activa por flanco de subida (en su representación síncrona). Es importante saber que en este caso la entrada SET pone a 1 la salida directa Q y la entrada *reset* la pone a 0.

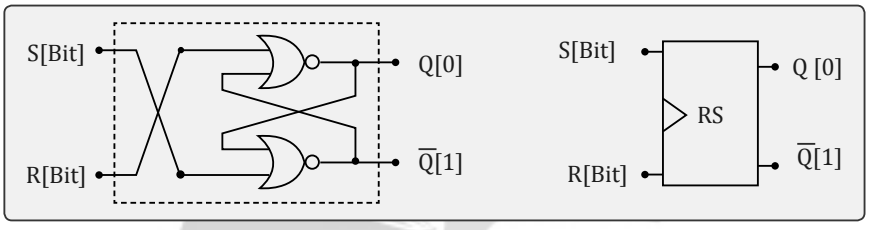


Figura 6.17 Esquema de representación de un *flip flop* RS activo al nivel alto

La tabla de verdad de un biestable RS asíncrono activo al nivel alto es la siguiente:

Entradas		Salida	Significado
R	**S**	**Q (t + 1)**	
0	0	Qt	Estado anterior
0	1	1	Puesta a Uno
1	0	0	Puesta a Cero
1	1	X	No permitido

Su expresión característica es: **Q(t + 1) = S + R' Qt**.

Como se puede apreciar, en una compuerta NOR la salida es forzada a 0 cuando una de sus entradas es 1, por ello, cuando R y S valen 1 en conjunto, ambas salidas (Q y Q') son 0; lo cual contradice el postulado del álgebra de Boole en el que una variable y su complementada no pueden tomar el mismo valor. Por ello, la combinación R=1 y S=1 no se permite en este biestable.

b. Biestable RS asíncrono (activo al nivel bajo): Este dispone de dos entradas denominadas R' y S' y dos salidas Q y Q', complementadas (directa y negada). La entrada S', cuando toma el valor 0, pone a 1 la salida directa Q, y la entrada R', cuando toma el valor 0, la pone a 0.

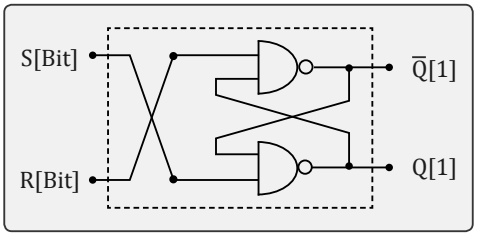


Figura 6.18 Esquema de representación de un *flip flop* RS activo al nivel bajo

La tabla de verdad de un biestable RS asíncrono activo al nivel bajo es la siguiente:

Entradas		Salida	Significado
R'	S'	Q (t + 1)	
0	0	X	No permitido
0	1	0	Puesta a cero
1	0	1	Puesta a uno
1	1	Qt	Estado anterior

Como se puede apreciar en una compuerta NAND la salida es forzada a 1 cuando una de sus entradas es 0, por ello, cuando R' = 0 y S' = 0 las salidas (Q y Q') son 1, lo cual contradice el postulado del álgebra de Boole en el que una variable y su complementada no pueden tomar el mismo valor. Por ello, la combinación R'= 0 y S' = 0 no se permite en este biestable.

c. Biestable JK: Este tipo de biestable es considerado como biestable universal. Dispone de tres entradas síncronas J y K para especificar la operación y CLK para disparar el biestable. También, consta de dos entradas asíncronas PR (PRESET) y CLR (CLEAR) y, por supuesto, dos salidas complementarias.

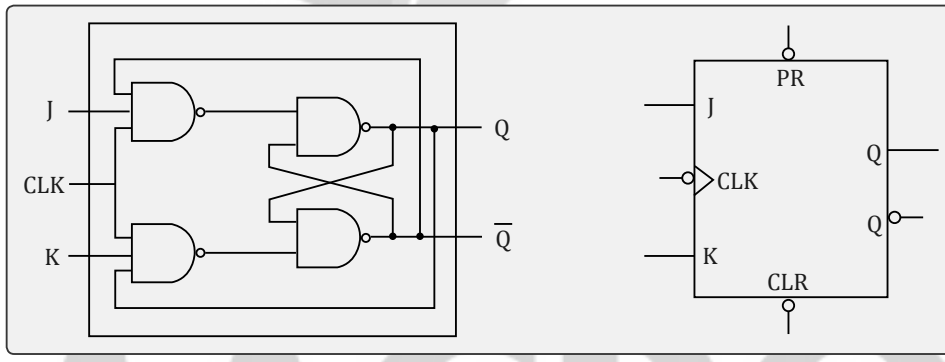


Figura 6.19 Esquema de representación de un *flip flop* JK

La tabla de verdad de un biestable JK asíncrono es la siguiente:

Entradas		Salida	Significado
J	K	Q (t + 1)	
0	0	Qt	Estado anterior
0	1	0	Puesta a Cero
1	0	1	Puesta a Uno
1	1	(Qt)'	Neg_Edo. anterior

Su expresión característica es: **Q(t+1) = JQt' + K'Qt**.

d. Biestable tipo D (DATA) activo por nivel: Es conocido como *latch* o cerrojo. Se trata de otro tipo de biestable conformado por una entrada D (datos) y dos salidas de estados complementarias (Q). Cuenta, además, con una entrada de reloj (CLK) activada por flanco de subida. También, puede contar con dos entradas más, conocidas por PR (PRESET) y CLR (CLEAR). Estas últimas son de tipo asíncrono. Un biestable D puede funcionar tanto de manera síncrona como de manera asíncrona (usa las señales PR Y CLR).

En este tipo de biestables cuando la señal de reloj es activa (nivel alto del reloj) la salida toma el valor de la entrada D y cuando la señal de reloj no es activa (nivel bajo del reloj) la salida permanece invariable (memorizada). Para la construcción de este biestable se parte del biestable R-S síncrono con la entrada R, la inversión de D y la entrada S igual a D. De esta forma R y S no pueden ser nunca iguales, evitando el estado no permitido del biestable RS.

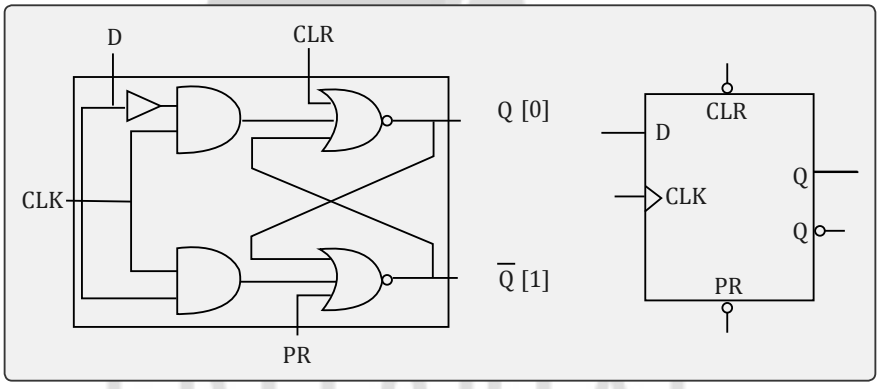


Figura 6.20 Esquema de representación de un *flip flop* tipo D activo por nivel

La tabla de verdad de un biestable tipo D activo por nivel es la siguiente:

Entradas				Salida	Significado	Tipo
PR	CLR	C	D	**Q (t + 1)**		
1	0	X	X	1	Puesta a Uno	Salida asíncrona
0	1	X	X	0	Puesta a Cero	Salida asíncrona
1	1	X	X	0	No permitido	Salida asíncrona
0	0	0	X	Qt	Estado anterior	Salida síncrona
0	0	1	0	0	Puesta a Cero	Salida síncrona
0	0	1	1	1	Puesta a Uno	Salida síncrona

Su expresión característica es: **D = Q (t + 1)**.

e. Biestable tipo T (Toggle): Consiste en un *flip flop* que se comporta como un biestable JK, en el que se unen las entradas J y K. Su representación síncrona es muy simple. Pues, en este tipo de *flip flop* cuando la señal de reloj es activa (flanco de bajada del reloj), si T vale igual a 0, las salidas permanecen iguales a su estado anterior, pero si T vale 1, las salidas cambian a su valor complementado. Cuando la señal del reloj no es activa, la salida permanece invariable (memorizada).

Se puede realizar este biestable a partir del biestable J - K síncrono con las entradas J y K unidas e iguales a T. De esta forma J y K no pueden ser nunca distintas, eliminándose esta posibilidad de la tabla de verdad del biestable J - K.

Su expresión característica es: **Q (t+1) = TQt' + T'Qt.**

La tabla de verdad de un biestable tipo T síncrono es la siguiente:

Entradas					Salida	Significado
(CL)'	(PR)'	(CK)'	J=T	K=T	Q (t + 1)	
1	0	X	X	X	1	Puesta a Uno
0	1	X	X	X	0	Puesta a Cero
0	0	X	X	X	1	No permitido
1	1	X	X	X	Qt	Estado anterior
1	1	X	0	0	Qt	Estado anterior
1	1	X	0	1	0	Puesta a Cero

Como se ha visto hasta el momento el manejo de biestables le permite comprender la forma en que permanecen registrados en memoria los datos introducidos en un circuito. Son muy útiles para comprender algunos puntos relacionados con microcontroladores y plataformas de hardware libre como Arduino.

B. Contadores

Un contador se define como todo circuito o dispositivo que genera una secuencia de valores que representan el número de pulsos de entrada contados. Son una serie de combinaciones sincronizadas por una señal de reloj externa. Con la combinación de flip flops, se pueden obtener contadores digitales, así como permitir la generación de registros. Los contadores digitales poseen varias clasificaciones:

Según su comportamiento con la señal de reloj son contadores asíncronos y síncronos.

El contador asíncrono está formado, en principio, por flip flops y lógica combinatoria adicional. Se llaman así, ya que la señal externa de reloj, en general, se conecta a la entrada de un solo flip flops y se propaga, luego, internamente. La ventaja es su sencillez. Su principal desventaja es su limitada velocidad de respuesta que depende de la cantidad de bits que maneje.

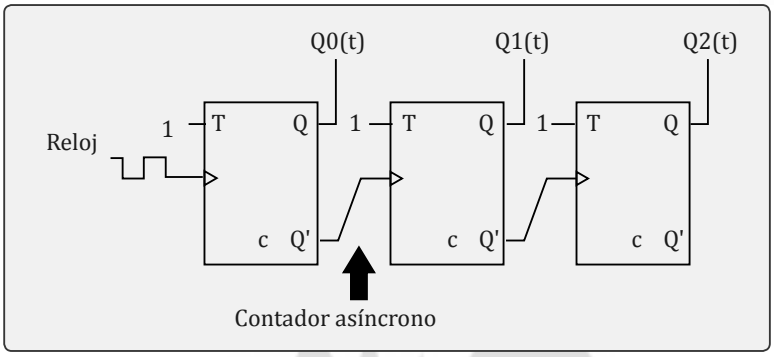


Figura 6.21 Esquema de conexión asíncrona de tres *flip flops*

Algunos circuitos integrados de este tipo son: CI 7490 (divisor por 2 y 5), CI 74196 (divisor entre 2 y 5), CI 74197 (divisor entre 2 y 8), CI 74393 (contador binario de 4 bits), etc.

El **contador síncrono** está formado, en principio, por *flip flops* y lógica combinatoria adicional. Se llaman así, ya que la señal externa de reloj, en general, se conecta a las entradas de reloj de todos los *flip flops*, simultáneamente. La ventaja es su mayor velocidad de respuesta respecto al asincrónico. Su relativa desventaja es su complejidad de conexión y alto consumo de energía.

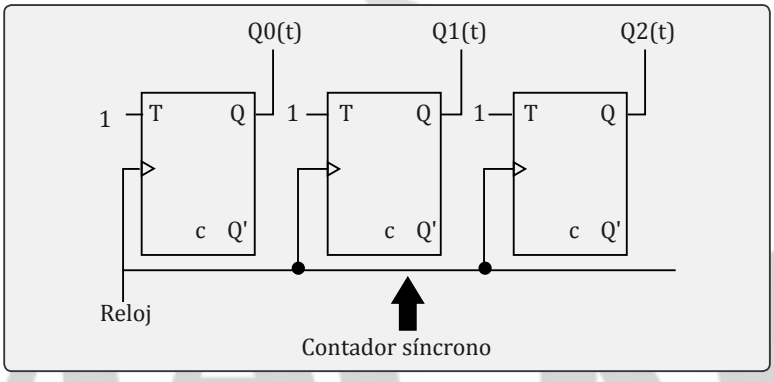


Figura 6.22 Esquema de conexión síncrona de tres *flip flops*

Algunos circuitos integrados de este tipo son: CI 74160A (módulo 10, *reset* asíncrono), CI 7491 (módulo 16), CI 74192 (bidireccional BCD), etc.

- **Según el formato de salida del conteo binario, BCD, arbitrario:** Por lo general, existen contadores digitales binarios y decimales. Un ejemplo de contador binario es el CI 74191, un BCD es CI 74192, y un ejemplo de contador decimal es el CI 74190.
- **Según el sentido de conteo ascendente o descendente:** Por lo general, algunos contadores incorporan entradas de control (S0 y S1), las cuales determinan la dirección del flujo. Los bits pueden desplazarse a la derecha o a la izquierda, según sea el caso.

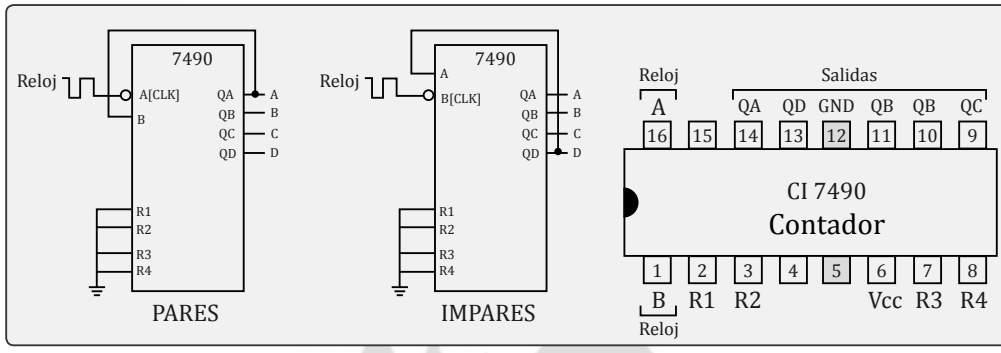


Figura 6.23 Esquemas de conexión de un circuito contador asíncrono 7490

El conteo que efectúan ciertos contadores digitales puede variar de acuerdo a su tipo de conexión, por ejemplo, mostrar un conteo de números pares o impares (de 0 a 9 para contadores de 4 bits).

A veces, muchos de los circuitos combinacionales o secuenciales (CI), como el caso de los contadores, se conectan a un componente que permite la visualización de los datos. El *display* de 7 segmentos es una buena solución, y a veces es recomendable el uso de dos o más en caso de requerir una salida de varios dígitos. Estos, por lo general, requieren del uso de decodificadores (en caso de convertir valores binarios a decimal).

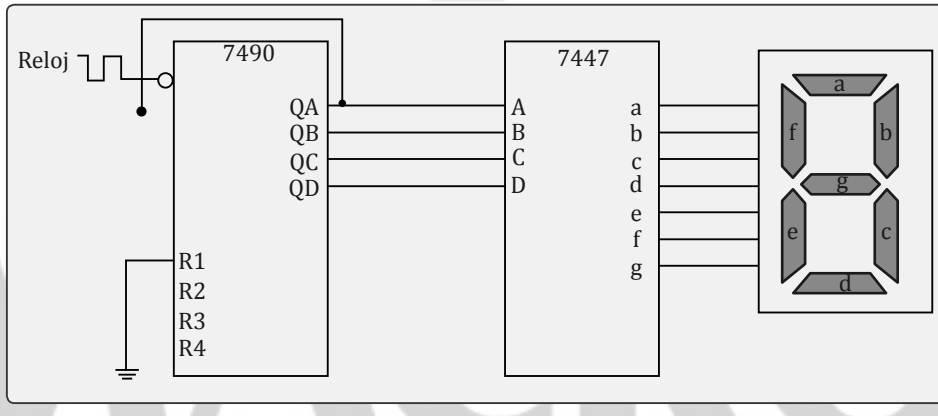


Figura 6.24 Datos de salida de un contador digital hacia un *display* de 7 segmentos

ACTIVIDAD 3

1. Monte sobre el constructor y simulador de circuitos digitales el esquema de conexión del circuito 7490 (contador de 4 bits).
2. Construya la tabla de verdad de dicho circuito y explicar su funcionamiento.
3. Monte el mismo circuito en protoboard físico.
4. Haga el montaje del circuito anterior con el uso de un *display* de 7 segmentos.
5. Realice el montaje de un contador de 0 a 99.

C. Registros

Un registro es un grupo o arreglo de *flip flops* manejado por una señal de reloj común. Es, básicamente, una unidad de almacenamiento. Existen registros con *latches*, *flip flops* y de desplazamiento. Entre las funciones de un registro se encuentran:

- Almacenamiento de datos.
- Conversión de datos de una forma a otra.
- Puede utilizarse como un contador especial.
- Manipulación de datos (registro de desplazamiento).

a. Registro de desplazamiento: En este apartado se realiza el estudio de los registros de desplazamiento, los cuales se definen como circuitos, donde la información de entrada se va desplazando en las salidas cuando se les aplica una señal de sincronismo; si se deja de enviar la señal de sincronismo, en la salida se mantiene la información anteriormente presente (memorizado).

La información, tanto de entrada como de salida, se puede introducir en serie o en paralelo. De acuerdo a lo anterior se obtienen diferentes tipos de registros como entrada/salida serie, entrada/salida paralelo, entrada paralelo/ salida serie, entrada paralelo/salida paralelo. Además de estos tipos hay registros de varios tipos de entradas y salidas de tratamiento de la información a la vez.

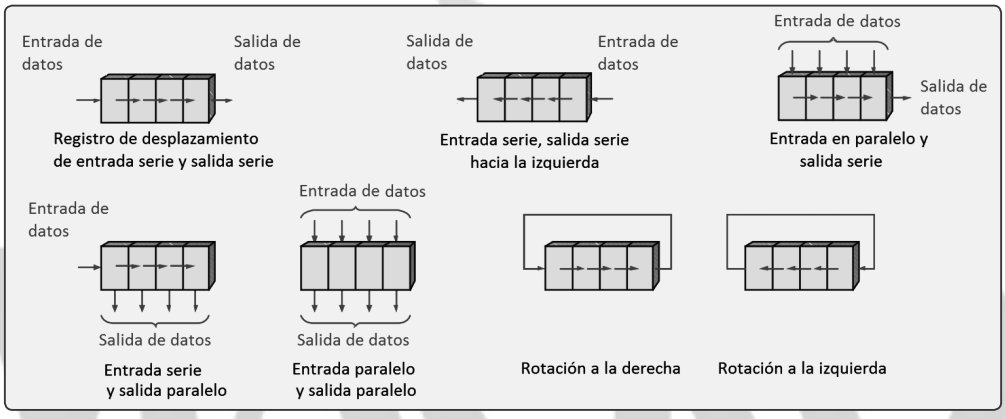


Figura 6.25 Esquema de flujo de entrada y salida de datos en un registro de desplazamiento

La capacidad lógica secuencial de los registros de desplazamiento permite a las computadoras enviar, almacenar y recibir datos binarios. Uno de los usos principales de estos registros es el almacenamiento de datos. Todos los sistemas de computadoras utilizan registros para almacenamiento de datos temporales con duración de microsegundos.

En una PC típica se encuentran 8, 16 o 32 *flip flops* conectados para formar registros, por ejemplo *flip flops* tipo D y JK por lo general.

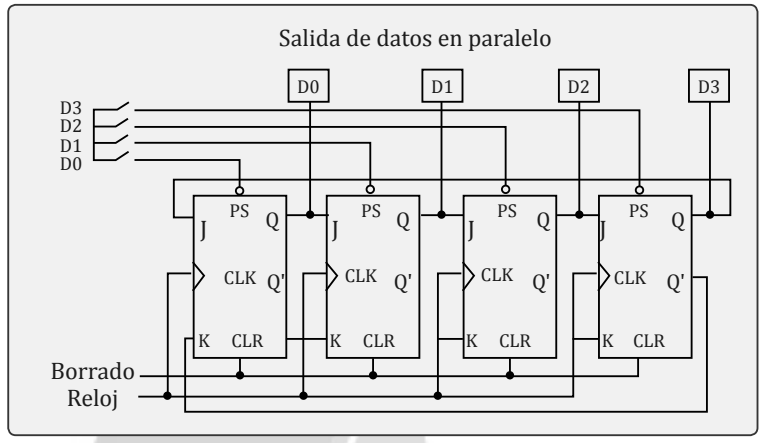


Figura 6.26 Registro de desplazamiento con *flip flops* (carga en paralelo)

Algunos de los circuitos integrados más conocidos para la implementación de proyectos, que impliquen el manejo de registros de desplazamiento (*flip flops*, con tecnología TTL o CMOS), son el CI 74HC164A (entrada serie, salida serie), el CI 74HC165 (8 bits, paralelo a serial), el CI 74HC166 (entrada paralelo, salida serie), el CI 74HC194A (bidireccional de 4 bits) y el CI 74HC195A (de 4 bits, universal).

El circuito 74194 (TTL) es, por lo general, conocido como registro de desplazamiento universal, sobre el cual se manejan 4 bits. Este registro posee 10 entradas y cuatro salidas, las cuales están conectadas a las salidas Q de cada *flip flop* del circuito.

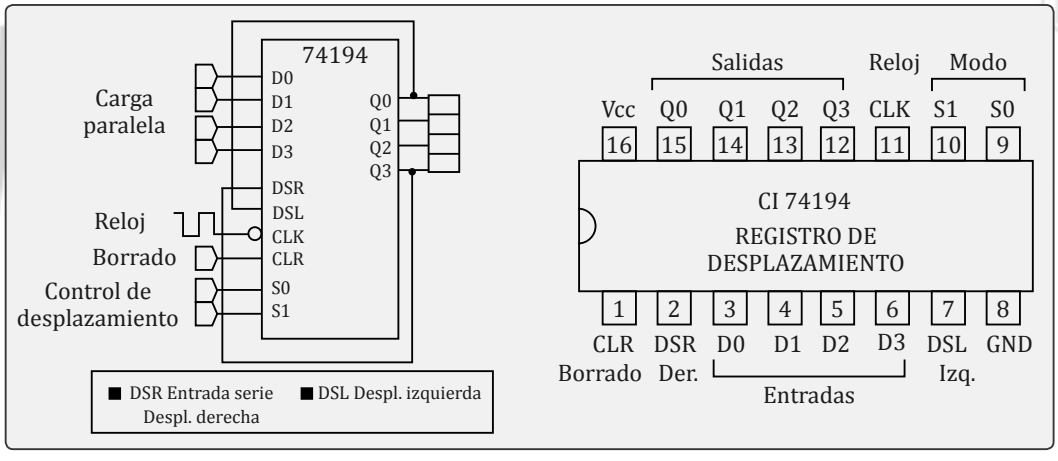


Figura 6.27 Esquema de distribución de pines del CI74194

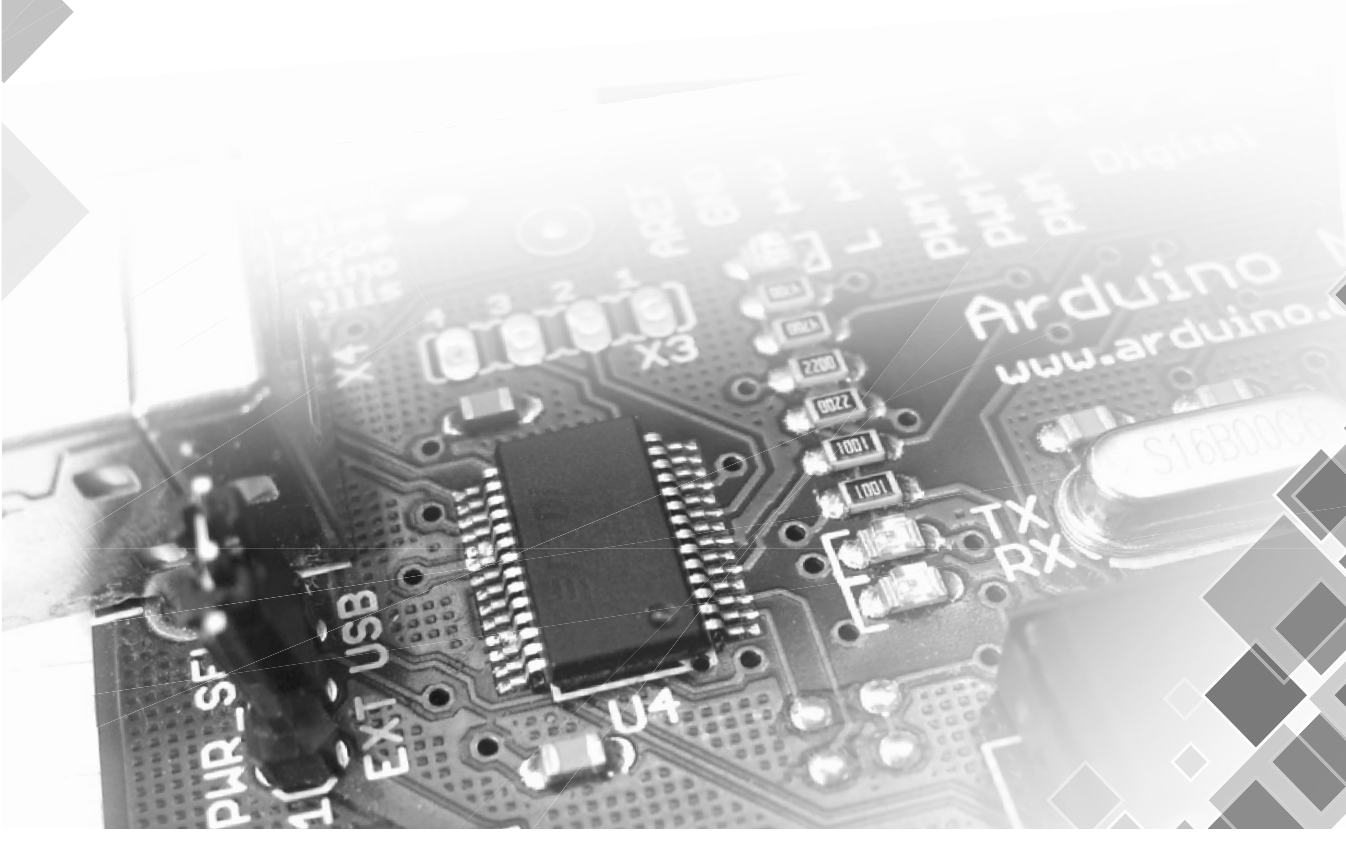
ACTIVIDAD 4

1. Monte sobre el constructor y simulador de circuitos digitales el esquema de conexión del circuito 74194 (registro de desplazamiento universal de 4 bits).
2. Construya la tabla de verdad de dicho circuito y explique su funcionamiento.
3. Monte el mismo circuito en protoboard físico.
4. Mencione por lo menos el nombre de tres circuitos de registro de desplazamiento TTL y tres CMOS.

En este Capítulo, se han descrito los principales circuitos digitales empleados en la electrónica digital y la ingeniería de hardware. Comenzando con temas como sistemas BCD y el uso de displays. Se han ilustrado también los circuitos combinacionales y los circuitos secuenciales.

Capítulo 7

Prácticas con Arduino



EDITORIAL
MACRO®

Como se ha mencionado, Arduino es una plataforma de harware y software libre, la cual ha tenido mucho auge en los últimos años. Sobre todo, porque simplifica el trabajo de índole electrónico que desde hace tiempo ha sido largo, tedioso y complicado. La electrónica, ha dado un salto enorme el cual se ve reflejado en plataformas que permiten la convergencia e impulsan proyectos de interés común a gran escala. En este capítulo se darán a conocer algunos datos interesantes sobre la plataforma Arduino, se expondrán algunos ejemplos de su programación y algunos ejercicios y prácticas que le servirán para introducirse al mágico mundo de la electrónica y la ingeniería de hardware.

7.1 INTRODUCCIÓN A ARDUINO

Arduino actualmente cuenta con una gran gama de productos oficiales entre los que se encuentran placas, módulos, *shields*, kits y accesorios.

Los diferentes productos que forman parte de la gama de esta plataforma, por lo general se encuentran clasificados, aunque se puede destacar el uso de Arduino UNO, Arduino Mega, Arduino Nano, Arduino Micro, Arduino Pro, etc. Sin mencionar algunas *shields* o módulos que son de gran ayuda al momento de integrar un proyecto. Para obtener mayor información se recomienda consultar la página principal de Arduino (www.arduino.cc).

ENTRY LEVEL	ARDUINO UNO, ARDUINO 101, ARDUINO PRO, ARDUINO PRO MINI, ARDUINO MICRO, ARDUINO NANO, ARDUINO STARTER KIT, ARDUINO BASIC KIT, ARDUINO MOTOR SHIELD
ENHANCED FEATURES	ARDUINO MEGA, ARDUINO ZERO, ARDUINO DUE, ARDUINO PROTO SHIELD
INTERNET OF THINGS	ARDUINO YÚN, ARDUINO MKR1000, ARDUINO ETHERNET SHIELD, ARDUINO GSM SHIELD, ARDUINO WIFI SHIELD 101
WEARABLE	ARDUINO GEMMA, LILYPAD ARDUINO USB, LILYPAD ARDUINO MAIN BOARD, LILYPAD ARDUINO SIMPLE, LILYPAD ARDUINO SIMPLE SNAP
3D PRINTING	MATERIA 101

Figura 7.1 Clasificación de los productos de Arduino según su página oficial

La placa de Arduino se presenta, originalmente, como una PCB de color azul, la cual se constituye por encima de otros componentes, de un microcontrolador. En sus inicios por el ATMega328/p, el cual ha ido evolucionando hasta convertirse en un chip de menor tamaño al original (tal es el caso de Arduino UNO edición SMD). El microcontrolador es el elemento central encargado de abastecer de numerosas funciones a la placa. Este

capítulo está enfocado en un estudio sobre Arduino UNO, sus componentes, su configuración, programación y prueba. Más adelante se describe su función como plataforma para la construcción de *sketch* o programas.[1]

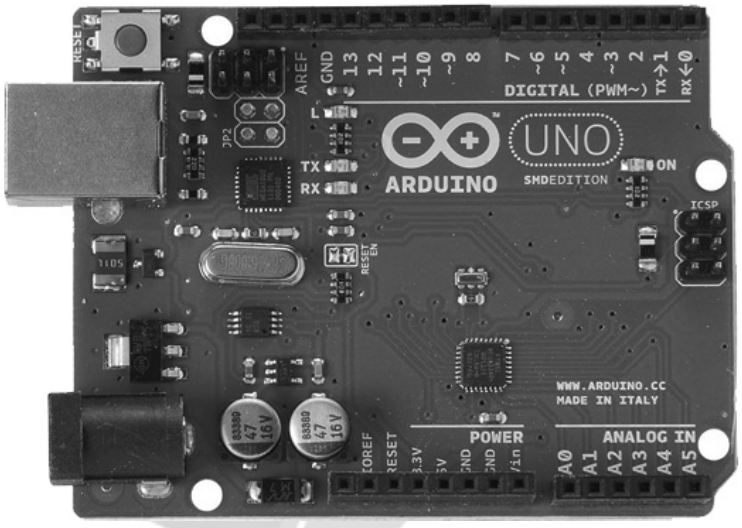


Figura 7.2 Interfaz de la placa de Arduino UNO edición SMD

El Arduino UNO tradicional se encuentra integrado por chip de 28 pines (de los cuales del 0 - 13 están rotulados como entradas digitales, del A0 al A5 como entradas analógicas y el resto para polarización o referencia). En la siguiente figura se muestra un esquema de disposición de pines del microcontrolador en función con la placa o de manera independiente. Lo último quiere decir que con mínimos conocimientos sobre electrónica, se puede montar un microcontrolador ATMega328 sobre protoboard y tratar de conseguir una configuración apropiada del Arduino para su prueba.

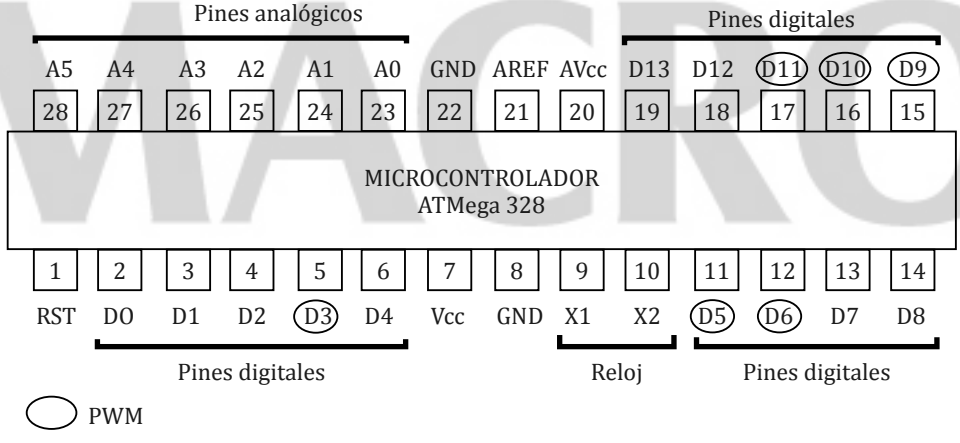


Figura 7.3 Esquema de disposición de pines del chip ATMega328 de Arduino UNO

[1] Para mayor información sobre descripción de la interfaz de Arduino UNO se sugiere consultar el capítulo uno de este libro.

7.2 CONEXIÓN Y CONFIGURACIÓN

Prácticamente, cualquier placa de Arduino permite su configuración y programación. Lo anterior por tratarse de hardware y software libre. Por lo tanto, para que la placa funcione sobre cualquier PC debe instalarse el driver o controlador correspondiente (debido a que la conexión se hace vía USB), el cual se puede descargar en conjunto con otros archivos desde la página oficial de Arduino. Estos están almacenados en una carpeta comprimida y disponible en la pestaña *Downloads* del sitio. El enlace aparece con el nombre Windows ZIP *file for non admin install*. En caso de querer realizar una instalación automática de Arduino sobre la PC se recomienda dar clic en el enlace Windows Installer.

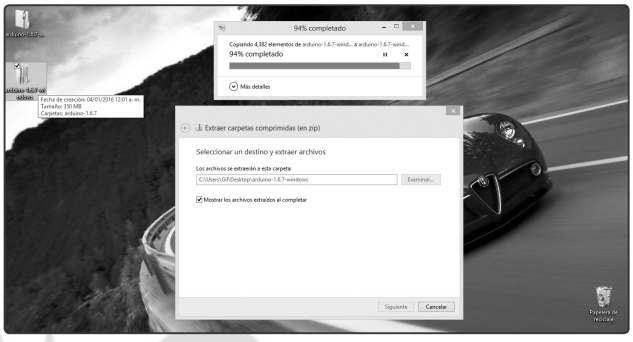
La carpeta comprimida es de suma importancia a la hora de trabajar con la plataforma, pues con ello se conseguirá la comunicación, la programación y la prueba de prototipos o proyectos. Esta carpeta, por lo general, incluye librerías, ficheros propios del hardware (AVR), ejemplos, *drivers*, complementos y desde luego el entorno de desarrollo o IDE para la programación (bajo lenguaje C). Arduino se encuentra disponible para los sistemas Windows, GNU-Linux y MAC-OS, aunque en este libro se realizará la explicación únicamente para el sistema operativo Windows.

Figura 7.4 Contenido de la pestaña Descargas del sitio oficial de Arduino

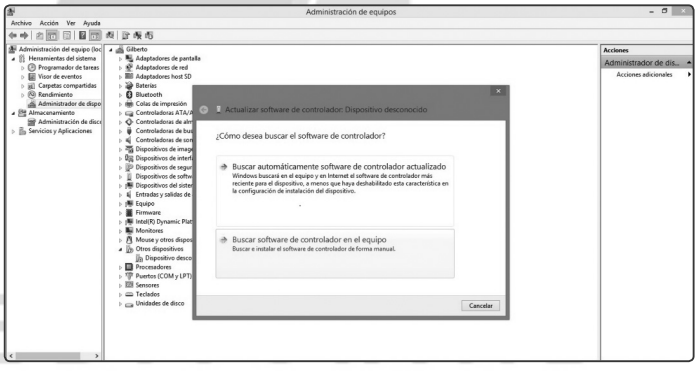
Para comenzar a trabajar con la placa de Arduino, esta debe conectarse a la PC vía USB. De esta manera, se suministra un voltaje no mayor a 5 volts, tensión suficiente para comenzar a trabajar. Posteriormente, la PC solicitará al usuario la carga de un controlador para ser reconocida y comience a funcionar.

Al conectar el dispositivo, debe cerciorarse del encendido del led ON (color verde) y del led de prueba número 13 de la placa (color naranja). Lo anterior es un buen síntoma de que todo funciona correctamente. Por igual debe analizar su reconocimiento desde la PC verificando el administrador de dispositivos. Si se ha optado por descargar la carpeta comprimida (Windows ZIP *file for non admin install*) desde el sitio de Arduino, quiere decir que desea realizar una instalación manual, en caso de haber elegido la opción Windows Installer, no será necesario efectuar los siguientes pasos, pues el *driver* será levantado automáticamente. En seguida se hace la explicación de la instalación manual del *driver* de Arduino UNO:

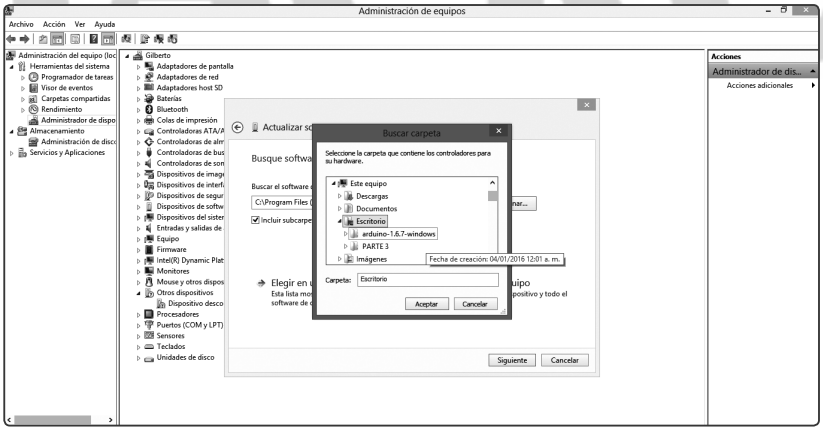
1. Descargue el paquete correspondiente o carpeta comprimida con los archivos necesarios de trabajo para la placa Arduino. Se recomienda que descomprima dicha carpeta y almacene la información en alguna ubicación dentro del equipo (por ejemplo, en el escritorio).



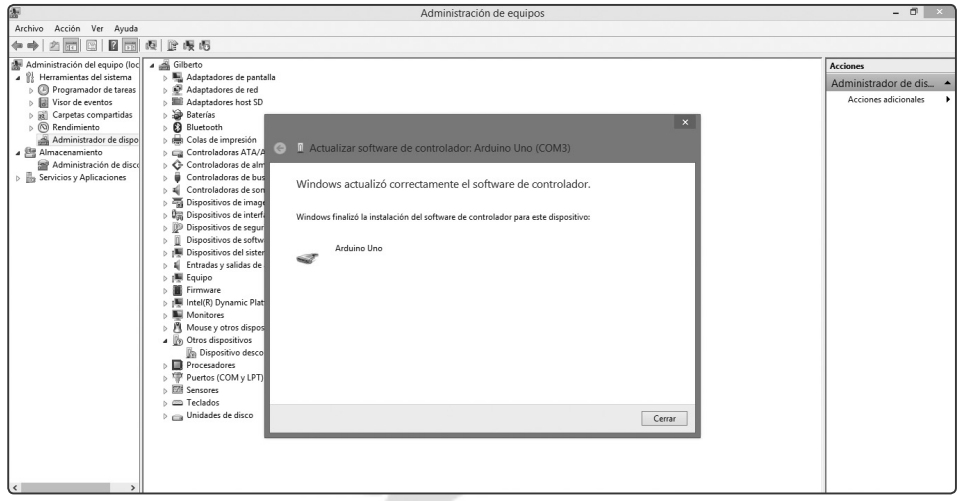
2. Después debe conectarse la placa a la PC (vía USB). Se nota que, por defecto, se solicita la actualización del controlador del nuevo dispositivo (el cual se señala como dispositivo desconocido) desde el administrador de dispositivos de la PC. Al dar clic derecho deberá elegir la opción **Actualizar software** de controlador. De este modo se procede a su búsqueda del *driver* en el equipo u otra ubicación.



3. Se procede a la búsqueda de software de controlador en el equipo (segunda opción). Como podrá notar, el equipo solicita la ubicación del driver (almacenado en la carpeta Windows ZIP previamente descomprimida en el escritorio). Para colocar la dirección exacta, debe dar clic sobre el botón **Examinar** y después elija la ruta correspondiente. Para terminar, oprima el botón **Aceptar** y después el botón **Siguiente**.

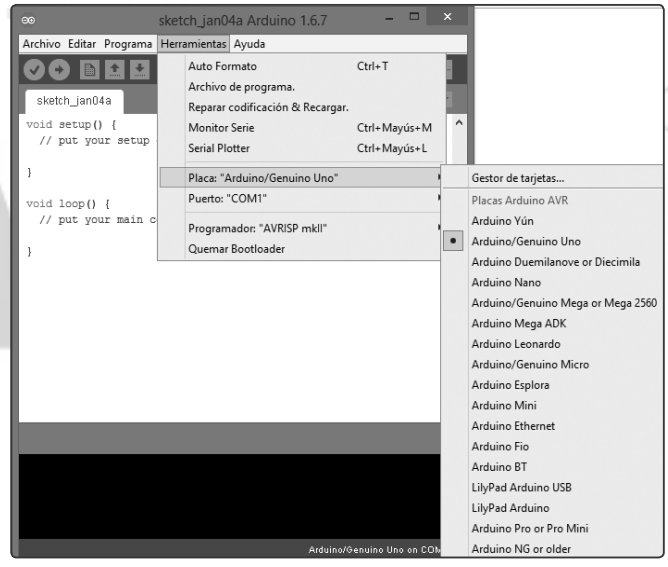


4. Realizado lo anterior, el sistema mostrará un cuadro de seguridad de Windows en el que solicita al usuario la autorización para llevar a cabo la instalación del software del dispositivo elegido. Oprima **Instalar** y espere. Al término de unos segundos aparecerá un nuevo cuadro en pantalla que indica que la instalación del software del dispositivo se ha llevado a cabo con éxito. Oprimir el botón **Cerrar**.



A partir de este momento, la base de datos del administrador de dispositivos queda actualizada con el nuevo dispositivo instalado. Ahora solo resta cerrar la ventana del administrador de dispositivos.

5. Ahora se procede a la carga del IDE de Arduino para la configuración y programación de la placa. Para ello debe abrir el archivo Arduino.exe (Aplicación) ubicado en el interior de la carpeta descomprimida sobre el escritorio. Al hacer doble clic aparece la ventana principal del entorno de desarrollo.



Una vez abierta la interfaz, se recomienda verificar la carga tanto de la placa de Arduino correspondiente como la habilitación de un puerto de la PC (COM).

7.3 DE LA ELECTRÓNICA A LA PROGRAMACIÓN

Hasta ahora se ha explicado en gran parte los aspectos más importantes de la electrónica. Se ha hecho hincapié en el desarrollo de proyectos, los cuales implican el uso de la electrónica y el estudio de la ingeniería de hardware. Pero poco se ha analizado con respecto a la programación y el desarrollo. Es en este punto en que se discuten paradigmas que le permiten al usuario mezclar el hardware con el software, es decir, la electrónica en convergencia con la programación y el desarrollo.

7.3.1 Programación en C con Arduino

En ámbitos de programación es preciso resaltar la existencia de ciertos conceptos y definiciones. A continuación, se listan algunos conceptos que servirán al usuario de base para familiarizarse con la programación de Arduino:

a. **Entorno de desarrollo:** Es el espacio dispuesto para la codificación, compilación y ejecución de un programa.

b. **Compilación:** Es el proceso de verificación del código para la detección de errores. Estos son expuestos en la parte inferior del entorno de programación con el fin de ser depurados.

c. ***Sketch*:** Es el nombre que recibe un programa escrito sobre el IDE de la plataforma Arduino.

d. **Librería:** Se define como un conjunto de ficheros o funciones abstractas que son invocadas desde un programa (código fuente), con el fin de expandir sus funcionalidades. Estas librerías son declaradas al inicio de la codificación sobre el compilador o entorno de desarrollo y por lo general son definidas con la extensión h (punto h). Por ejemplo, #include <Firmata.h>, #include <Servo.h>, #include <conio.h>. La palabra «include» seguida del símbolo # es una forma estándar para incluir librerías en lenguaje C.

e. **Comentario:** Es todo aquel texto escrito dentro de un *sketch*, el cual no afecta la funcionalidad del programa. Existen dos formas de declarar un cometario: comentarios para una sola línea de código y comentario para varias líneas. La sintaxis para el primer caso es la siguiente: // comentario. En caso de abarcar más de una línea, se utilizan de la siguiente manera: /*comentario de varias líneas */.

f. **Identificador:** Se define como un conjunto de caracteres alfanuméricos de cualquier longitud que sirve para identificar las entidades del programa (clases, funciones, variables, constantes. Llámese variable y constante a un conjunto de datos que se almacenan en la memoria de una PC). Un identificador puede ser la combinación de letras y números. El lenguaje C establece algunas reglas y características para sus identificadores:

- Han de iniciarse con una vocal o consonante (mayúscula o minúscula). Aunque por convención, los nombres de variables empiezan por una letra minúscula y cuando están compuestos por más de una palabra, estas se colocan juntas y la siguiente palabra comienza con mayúsculas. Por ejemplo, “temRecamara”.

- No lleva caracteres especiales, ni caracteres en blanco.
- Tiende a estar asociado con el objeto (dato) al que se hace referencia en el programa.
- Deben ser breves (una palabra) y poseen un límite de máximo 31 caracteres.

g. Operador: Es definido como signo o signos que determinan la operación a realizarse entre dos o más valores. En programación existen por lo regular los siguientes operadores:

- Aritméticos: Ejemplo de ello son la suma (+), resta (–), multiplicación (*), módulo (%), división entera (/), potencia (**).
- Lógicos o booleanos: Son por lo general los operadores asociados a las compuertas lógicas AND (&&), OR (||) y NOT.
- Relacionales o de comparación: Ejemplo de ello son el signo de igual (=), diferente de (! =), menor qué (<), mayor qué (>), menor o igual (<=) y mayor o igual qué (> =).

h. Variable: Es aquel dato que cambia su valor durante la ejecución de un programa.

i. Constante: Es aquel dato que no cambia su valor durante la ejecución de un programa.

j. Tipo de dato: Hace referencia al tipo de valor que puede adoptar una variable o constante. En programación existen diferentes tipos de dato. Los más conocidos son:

Tabla 7.1 *Tipos de datos para lenguaje utilizado en Arduino*

Tipo de dato	Declaración	Descripción	Ejemplo
Entero	Int	Acepta valores numéricos de tipo entero (con signo)	-2, 201, 1, -1
Punto flotante	Float	Acepta valores numéricos de tipo real (valores que contienen una parte entera y una parte fraccionaria)	-3.9, 3.1416, 0.001
Carácter	Char	Acepta caracteres Unicode (código ASCII)	'A', 'b', 'X', '7', '/'
Cadena	String	Acepta un conjunto de caracteres (palabras o frases)	"Macro", "mi nombre", "Gilberto"
Booleano	Boolean	Acepta valores lógicos	0, 1, TRUE, FALSE

Existen otros valores con signo, sin signo (*unsigned*) e incluso valores como byte, short, double, word, array, etc.

Para obtener información con respecto a tipos de datos, variables, constantes, estructuras, interruptores, operadores para Arduino, se recomienda consultar la pestaña **Learnig** > Reference > **Language reference** del portal oficial de la plataforma.

k. **Macro:** Son rutinas (conjunto de instrucciones) que tienen como fin la simplificación de un programa (reducción de líneas de código). Son declaradas al inicio del programa (después de las librerías) de la siguiente forma: #define [macro]. A menudo son utilizadas para evitar escribir subrutinas en repetidas ocasiones y permitir así la redundancia de instrucciones.

l. **Función:** Es un módulo dentro de un programa que cumple con una tarea específica. Dentro de lenguaje C se pueden crear varias funciones, las cuales deberán ser definidas para poder ser invocadas. En Arduino por *default* se tienen dos funciones principales: void setup() y void loop(). Estas, generalmente, se declaran con un nombre seguido de la apertura y cierre de paréntesis (), sin punto y coma.

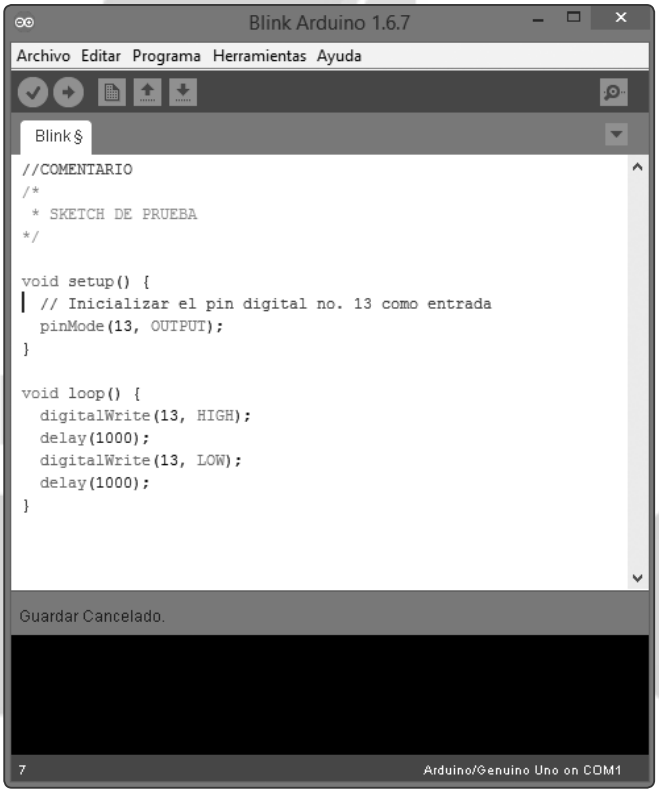


Figura 7.5 Las principales funciones del entorno de Arduino son void setup() y void loop()

Fuente: el autor.

Desde la función **void setup()** se definen los pines y se establece si fungirán como entradas o salidas digitales. En la función **void loop()**, por lo general, se colocan los patrones de repetición. Aunque, es común la declaración y juego de pines también. En esta área se colocan también las estructuras de control del programa.

m. **Estructura de control:** Se define como conjunto de sentencias que permiten el control de flujo de acciones en un programa. En programación existen, por lo regular, dos tipos de estructuras de control: selectiva y cíclica.

- **Selectivas:** Originalmente este tipo de estructura permite la evaluación de una condición. En su forma más básica (simple), si esta se cumple, se efectúa un número determinado de acciones, en caso contrario, el programa termina. No obstante, también se puede valorar más de una condición (mediante el uso de operadores relacionales o de comparación y operadores lógicos). De allí se desprenden la estructura selectiva doble y múltiple de las cuales se hablará más adelante.

  Las estructuras selectivas más comunes son *if* (si) en sus diferentes modalidades y la sentencia *switch-case* (según sea el caso), la cual emplea un determinado número de casos forzando así al usuario a seleccionar solo una de las múltiples opciones disponibles.

  La sintaxis de la sentencia *if* en su forma más simple es:

```
if (condición)
{ // Inicio de la estructura
   Acciones
} // Fin de la estructura
```

  Como se mencionó, una estructura selectiva, a menudo, se auxilia del uso de ciertos operadores, mismos que están dispuestos para jugar con las condiciones (realizar comparaciones y auxiliar en ocasiones al usuario en su selección). A continuación, se muestra un ejemplo del uso de operadores:

```
if (variableA > 100)
{
   digitalWrite(LedPin, HIGH); // el valor que ocupa HIGH, puede ser incluso LOW
}
```

  La sintaxis de las instrucciones dependerá siempre del lenguaje utilizado. En Arduino se utiliza la sentencia *digitalWrite* para escribir (asociar) un valor booleano (nivel de tensión) a un determinado pin digital.

  La estructura selectiva doble y múltiple, se presenta en la forma *if-else* (si-de lo contrario). A veces es empleada por programadores para jugar con más de una condición. La sintaxis es la siguiente:

```
if (condición)
{              Acción_A   }
else {
               Acción_B
         }
```

Un ejemplo de lo anterior sobre programación de C en Arduino se vería del siguiente modo:

```
if(digitalRead(8)==1)
{
    // Ejemplo de acciones
    digitalWrite (13, HIGH);
    delay(1000);
 }
else{
    digitalWrite (10, HIGH);
    delay(2000);
       }
```

Como se puede apreciar, sigue surgiendo un mayor número de palabras reservadas para este entorno. En esta ocasión se tiene la línea *digitalRead*. Esta, a diferencia de *digitalWrite*, permite la lectura del valor de un pin digital condicionando al programa a que, si este es igual a 1 (o nivel de tensión HIGH), se realice un determinado número de acciones (se encienda el pin número 13 por 1 segundo como lo estipula el *delay*). De lo contrario (*else*) se realizan las acciones comprendidas en el segundo grupo de llaves (lograr que se encienda el pin número 10 por dos segundos). La palabra «delay» significa 'retardo', este está definido en milisegundos (1 ms = 1000 uds).

La estructura selectiva múltiple de decisión exclusiva se implementa con la sentencia *switch-case*. Esta permite la elección de un solo caso de un menú de opciones. Su sintaxis es la siguiente:

```
switch (variable) {
  case 1:
    // acciones
    break;
  case 2:
    // acciones
    break;
  default:
    // acciones
  break;
}
```

La sentencia *break* se encarga de romper o finalizar un caso. Mientras que la sentencia *default* toma el valor por defecto que puede estar considerado o no entre los casos. Es importante notar que cada sentencia de control tiene un inicio y un fin, el cual se encuentra señalado con llaves {}.

- **Cíclicas:** Son estructuras que permiten la iteración de una o varias instrucciones en un determinado número de veces. Son, a menudo llamadas bucles. Existe la sentencia *for* (para), la sentencia *while* (mientras) y *do while* (hacer-mientras). Cada una de estas ha de ocuparse según las necesidades del usuario, puede incluso combinarlas entre sí.

  La estructura *for*, habitualmente, se utiliza para repetir un bloque de sentencias o instrucciones. Está integrada por una variable de inicio, una condición y un contador (incremento o decremento). La sentencia *for* es útil para cualquier operación repetitiva, y se usa a veces en combinación con arreglos. Su sintaxis es la siguiente:

```
for (inicialización; condición; contador)
{
    //acciones
}
```

  Es importante notar la existencia de las llaves de inicio y fin del bucle. Para la declaración de un ciclo *for* debe omitirse el punto y coma. A continuación, se muestra un ejemplo de su implementación en Arduino.

```
    for (int a = 2; a < 7; a++)
    {
      digitalWrite(a, HIGH);
      delay(100);
      digitalWrite (a, LOW);
      delay(100);
    }
```

  La sentencia *while* hace posible la repetición de un conjunto de acciones de forma continua e infinitamente, hasta que la expresión dentro del paréntesis, sea falsa. Su estructura es muy similar a la del *for*. Su sintaxis es la siguiente:

```
while(expresión o condición)
{
  // acciones
}
```

  Un ejemplo de aplicación de la sentencia *while* es la siguiente:

```
var = 0;
while(var < 100)
{
var++;
}
```

Antes de comenzar a declarar una sentencia *while*, es necesario efectuar la declaración de una variable de inicio. Después de esto, se escribe la sentencia *while* que contiene la expresión o condición (que implica el manejo de la variable inicial). Nótese la declaración del contador, el cual va incluido entre la apertura y cierre de llaves.

```
do
{
    // bloque de acciones
} while (prueba de la condición);
```

Un ejemplo de aplicación en Arduino es cuando se desea verificar la función de uno o más sensores por determinado tiempo.

```
do
{
  delay(100);
  x = readSensors();  // verificar sensores
} while (x < 300);
```

A. Primeros pasos

Ya que se conocen algunos términos que sin duda serán de utilidad para el usuario al momento de trabajar con un entorno de programación, se explicará la forma de programar, compilar, depurar y ejecutar un *sketch* capturado desde el IDE nativo de la plataforma Arduino. Para ello, es imprescindible conocer la función que tiene cada uno de los íconos de la barra de herramienta del entorno. En la siguiente infografía, se muestra la información necesaria:

Partes de la interfaz principal del IDE de Arduino

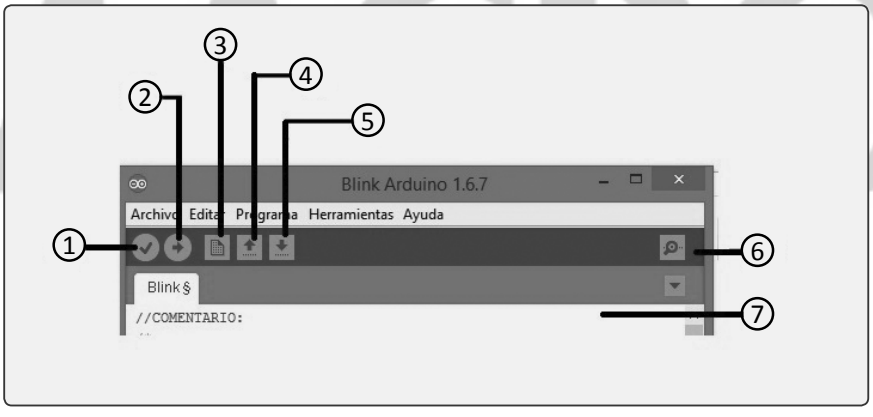


Figura 7.6 Descripción de la barra de herramientas de la ventana principal del IDE de Arduino
Fuente: el autor.

1. Verificar: Realiza la revisión del código en busca de errores. Si los hay, se pueden visualizar en la parte inferior izquierda de esta ventana, de lo contrario, la interfaz emitirá un mensaje de revisión exitosa y sin errores.
2. Subir: Permite vaciar el código generado desde el IDE hacia el microcontrolador de la placa de Arduino.
3. Nuevo: Permite la apertura de una nueva ventana sin tener que cerrar la actual.
4. Abrir: Permite la apertura de archivos (con extensión .ino) almacenados en unidad de disco.
5. Salvar: Guarda los cambios del archivo (codificación) actual. Generalmente, los almacena en una carpeta origen de Arduino con la opción de crear una ruta propia para almacenar los archivos generados.
6. Monitor serial: Muestra una ventana independiente que permite la interacción entre la placa y la PC. Esta ventana permite la inserción de comandos o, en su caso, la visualización de datos de salida.
7. Área de trabajo: Es el espacio para el desarrollo de sketches.

7.3.2 Electrónica con Arduino

El manejo de Arduino, habitualmente, implica el uso de placas de prueba para el montaje de prototipos o proyectos electrónicos a escala (como el caso del protoboard). Se emplean también esquemas de conexión, los cuales resultan de gran utilidad para el usuario que desea comprender el diseño de su implementación física. Son un recurso ideal que sirve de guía para la conexión real. Estos pueden ser trazados a mano o mediante alguna herramienta de simulación (por ejemplo, *Fritzing*).

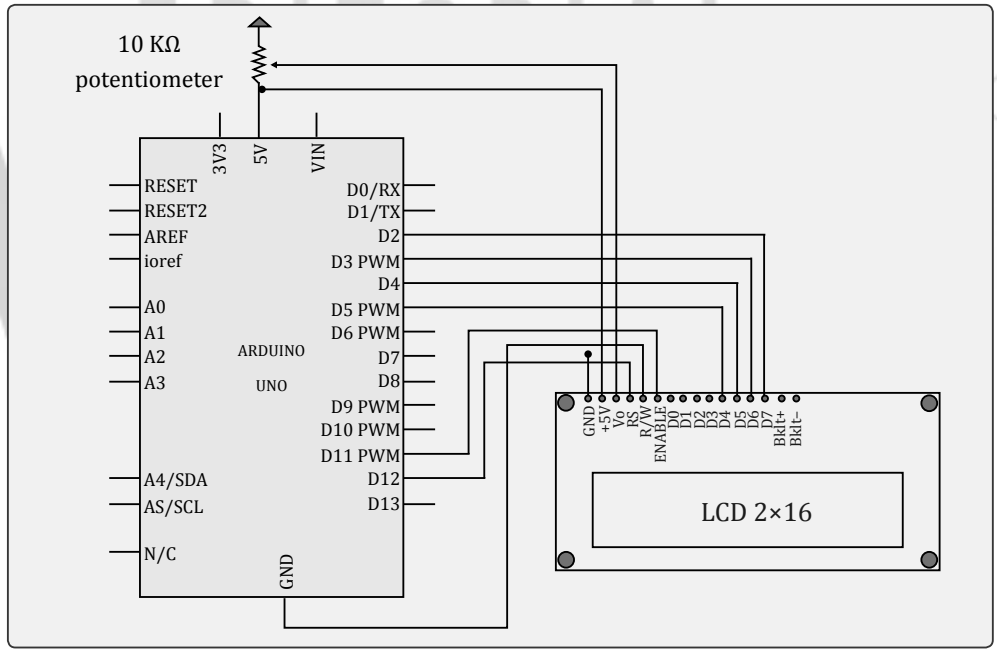


Figura 7.7 Conexión física de una patalla LCD 16 × 2 a la placa de Arduino UNO
Fuente: el autor.

Como se sabe, para cualquier montaje electrónico usualmente se emplean un conjunto de componentes electrónicos ideales para cubrir la expectativa del producto. Una vez montado el circuito, este es programado mediante el IDE de Arduino (en comunicación con el microcontrolador) para ejercer una función específica. En la imagen anterior se muestra la conexión de una pantalla LCD 16 × 2 a la placa de Arduino. Para esto serán necesarias las conexiones que se indican en el esquema. En primera instancia, la pantalla debe polarizarse para después conectar los pines digitales correspondientes (señalados con la letra D) a la placa de Arduino. En este caso se precisa la conexión de resistencias y del manejo de un potenciómetro de 10K ohmios para variar la luminosidad de la pantalla.

Para llevar a cabo cualquier conexión física, aparte del uso de protoboard, deben emplearse cables especiales (*jumper*), los cuales facilitan su inserción sobre los *headers* hembra u orificios dispuestos en la placa de Arduino. Debe tomarse también en cuenta la forma en que se deben conectar las resistencias (dos arreglos posibles). Estas pueden estar dispuestas tras una conexión Pull-Up o una conexión Pull-Down.

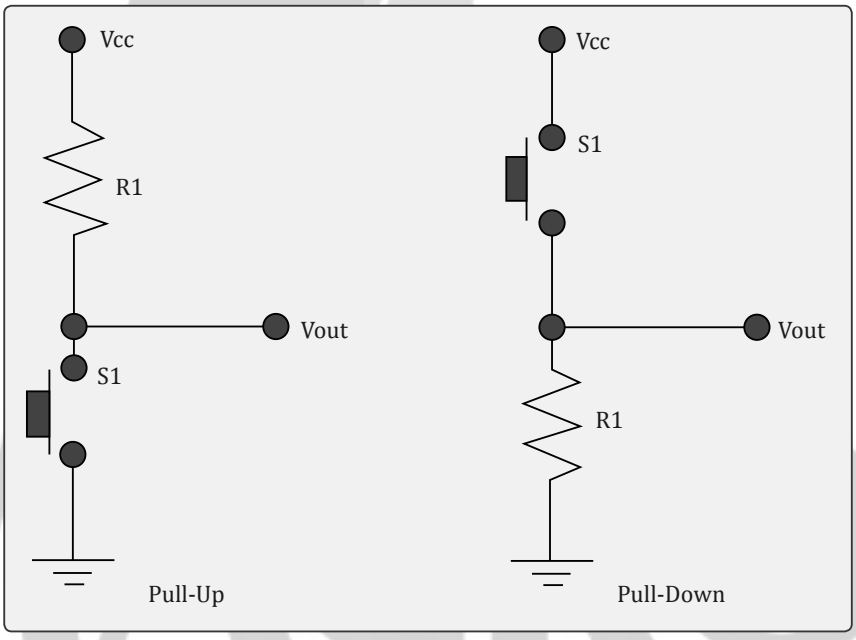


Figura 7.8 Esquema de conexión Pull-Up y Pull-Down
Fuente: el autor.

Como se aprecia en la figura anterior, ambos esquemas incluyen una resistencia (R1), un interruptor de pulsación (S1), una línea de voltaje de salida (*Vout*) y una conexión a VCC y a GND, respectivamente. Lo ideal será conectar sobre protoboard ambos esquemas de conexión para verificar su funcionamiento, ya que gracias a esto, se logra entender gran parte de la forma en que trabajan los circuitos que incluyen interruptores y líneas de salida de voltaje.

Para entender el funcionamiento de una conexión *Pull-Up*, se puede comparar con el funcionamiento de un retrete, en el que mientras no se oprima la palanca de descarga, el tanque se mantendrá lleno (HIGH). Una vez oprimida, el tanque de agua se quedará vacío (LOW). El funcionamiento de la conexión *Pull-Down* es más sencillo aún. Su funcionamiento es idéntico al de un timbre. Mientras el botón no se oprima, se obtiene un nivel de tensión bajo para *Vout*, de lo contrario, el nivel de tensión para *Vout* es alto.

Con lo antes visto, sobre programación y esquemas de conexión electrónica, se puede comenzar a incursionar con la plataforma Arduino. Para ello se plantea realizar un par de sketches básicos y su implementación electrónica:

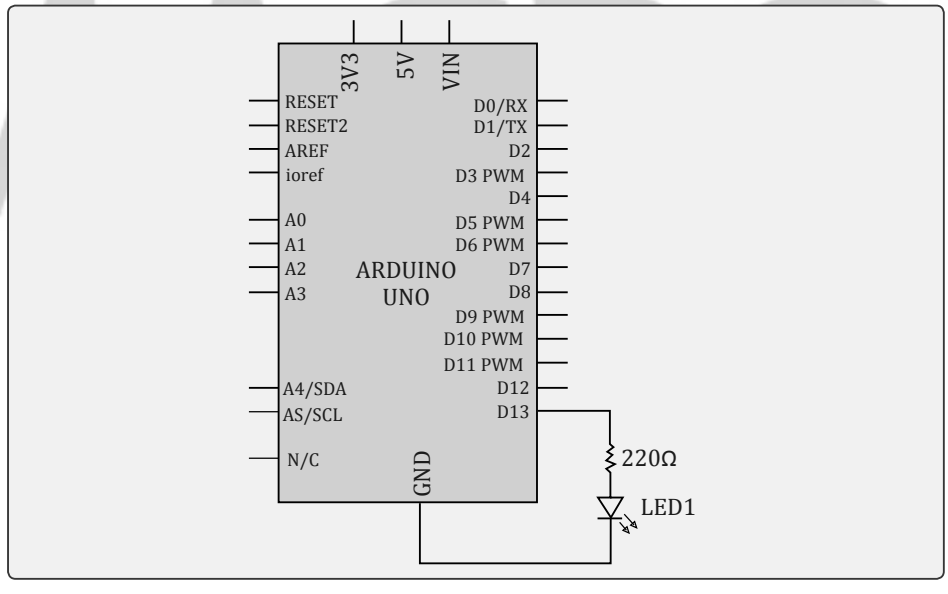
Ejemplo de codificación de *sketch* en Arduino

1. **ENUNCIADO:** Lograr el encendido y apagado de un led por un segundo. Se pretende desarrollar tanto la conexión física, como su *sketch*.

Sketch

```
Int led = 13;
void setup()
{
   pinMode(led, OUTPUT);
}
void loop()
{
   digitalWrite(led, HIGH);
   delay (1000);
   digitalWrite(led, LOW);
   delay (1000);
}
```

Conexión



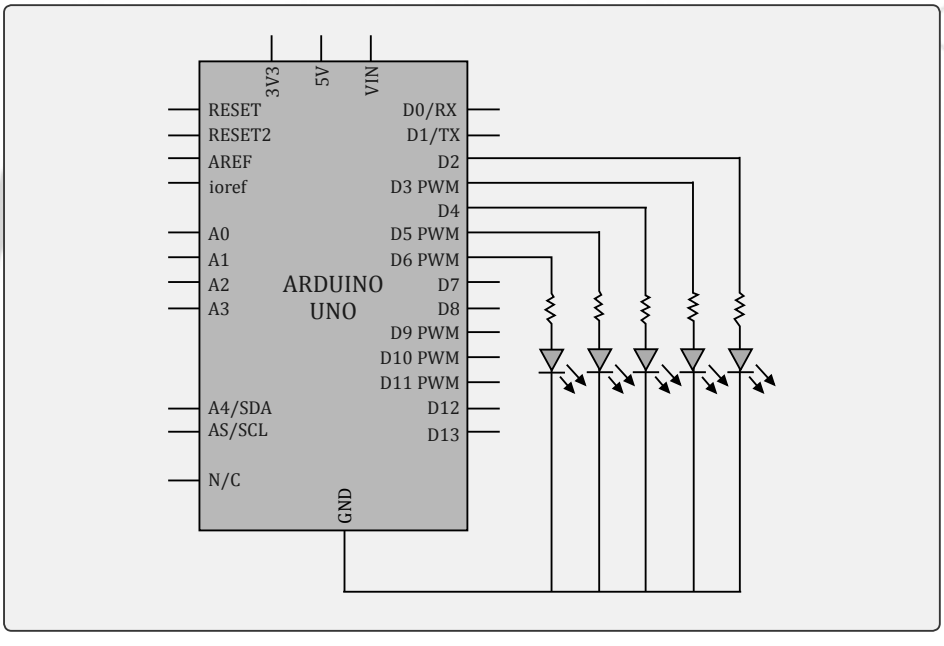
2. **ENUNCIADO:** Conseguir una secuencia de iluminación de cinco ledes (de manera ascendente y descendente) con el uso de la sentencia *for*.

   Se pretende desarrollar tanto la conexión física, como su *sketch*.

   Sketch

```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}
void loop()
{
    for (int a = 2;a < 7;a++)
    {
      digitalWrite(a, HIGH);
      delay(100);
      digitalWrite(a, LOW);
      delay(100);
    }
 for (int a = 5;a > 1;a--)
    {
      digitalWrite(a, HIGH);
      delay(100);
      digitalWrite(a, LOW);
      delay(100);
    }
```

   Conexión



7.4 EJERCICIOS CON ARDUINO

En vista de lo anterior, en esta sección se plantea comenzar a desarrollar programas que impliquen el uso de otros componentes electrónicos como LDR, pantallas LCD, sensores, teclados y servomotores.

7.4.1 Manejo de una LDR en Arduino

Hay que recordar que una LDR es un componente capaz de detectar la presencia y ausencia de luz. En el siguiente código se muestra la forma de interactuar con una fotorresistencia y un conjunto de ledes.

3. **ENUNCIADO:** Realizar un programa que logre encender todos los ledes y cuando se tapa la fotorresistencia se inicia una secuencia en orden descendente del pin 13 al 9 de la placa Arduino.

```
void setup()
{
pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
pinMode(11, OUTPUT);
pinMode(10, OUTPUT);
pinMode(9, OUTPUT);
pinMode(8, INPUT);
digitalWrite(13, HIGH); // enciende todos los ledes
digitalWrite(12, HIGH);
digitalWrite(11, HIGH);
digitalWrite(10, HIGH);
digitalWrite(9, HIGH);
delay(2000);
digitalWrite(13, LOW); // apaga todos los ledes
digitalWrite(12, LOW);
digitalWrite(11, LOW);
digitalWrite(10, LOW);
digitalWrite(9, LOW);
delay(2000);
digitalWrite(13, HIGH); // enciende todos los ledes
digitalWrite(12, HIGH);
digitalWrite(11, HIGH);
digitalWrite(10, HIGH);
digitalWrite(9, HIGH);
delay(2000);
digitalWrite(13, LOW); // apaga todos los ledes
  digitalWrite(12, LOW);
  digitalWrite(11, LOW);
  digitalWrite(10, LOW);
  digitalWrite(9, LOW);
  delay(2000);
}
```

```
void loop()
 {
     if(digitalRead(8)==1)
    {        // cuando se tapa la fotoresistencia inicia una secuencia en orden
descendente del 13 al 9
  digitalWrite(13, HIGH);
  delay(retardo);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  delay(retardo);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  delay(retardo);
  digitalWrite(11, LOW);
  digitalWrite(9, HIGH);
  delay(retardo);
  digitalWrite(9, LOW);
       }
            else  {
            digitalWrite(10, HIGH);
                      }
 }
```

ACTIVIDAD 1

1. Realice el esquema de conexión correspondiente al programa anterior (manejo de LDR).
2. Cablee de manera física el esquema desarrollado en el punto 1.
3. Consiga una secuencia en forma ascendente.

7.4.2 Manejo de una pantalla LCD 16 x 2 en Arduino

Una pantalla LCD 16 × 2, contiene 16 columnas y 2 filas. Son elementos que permiten la visualización de datos y a menudo pueden ser conectadas en conjunto con un potenciómetro, que hace posible la variación de luminosidad. En el siguiente código se muestra la forma de interactuar con una pantalla de este tipo en Arduino:

4. **ENUNCIADO:** En este ejercicio se muestra un texto en pantalla que dice: hola mundo.

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
  lcd.begin(16, 2);
  lcd.print("hola mundo!");
  delay(1500);
}
void loop()
{
  lcd.setCursor(0, 1);
//lcd.print(millis() / 1000);
  lcd.print("_");
}
```

ACTIVIDAD 2

De acuerdo al ejemplo anterior, realice lo siguiente:

a) Cablee de manera física el esquema de conexión de una pantalla LCD a la placa Arduino (el cual se muestra en la figura 7.7 de este capítulo).

b) ¿Cuáles son las funciones necesarias para desplazar el texto a modo de marquesina?

c) ¿Cómo se consigue un desplazamiento de derecha a izquierda y viceversa?

d) Imprima su nombre completo y desplazarlo hacia la izquierda a modo de marquesina.

7.4.3 Manejo del sensor de temperatura en Arduino a través del monitor serial

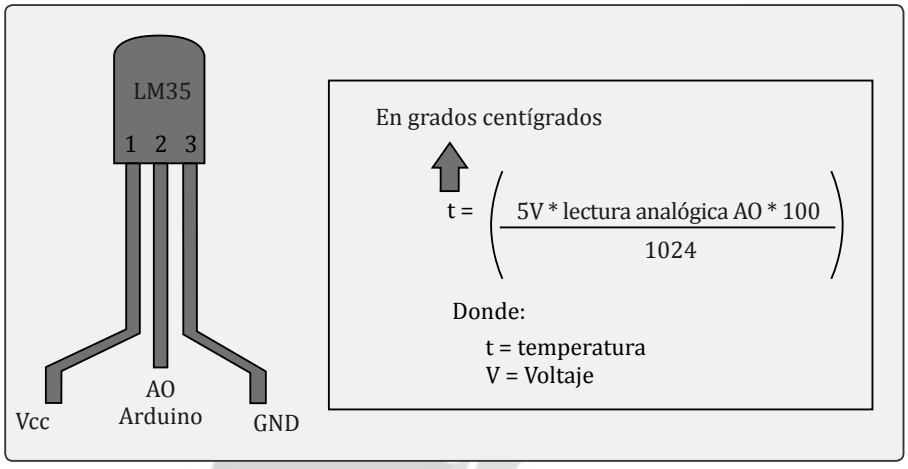
Un sensor de temperatura es capaz de arrojar datos que son detectados a través del medio ambiente. Estos datos a menudo se muestran en una pantalla *display* o, en su defecto, desde el monitor serial de la IDE de Arduino. En el siguiente código se muestra la forma de interactuar con un sensor LM35 a través del monitor serial:

5. ENUNCIADO: Efectuar la lectura de temperatura con el uso del sensor LM35. Uso del monitor serial.

```
float temp;
Int tempPin = 0;
void setup()
{
   Serial.begin(9600);
}
void loop()
{
   temp = analogRead(tempPin); // tempPin devuelve un valor digital de 0 a 1024
   temp = temp * 0.48828125;
   Serial.print("TEMPERATURA = ");
   Serial.print(temp);
   Serial.print("*C");
   Serial.println();
   delay (1000);
}
```

Para comprender mejor este código es necesario realizar las siguientes aclaraciones:

- Por cada °C, la tensión en *Vout* aumenta a 10Mv (milivoltio).
- Para contar con 5 V máximos, se necesitan 500° C
- La fórmula para el cálculo de la temperatura es:



Pudiendo entenderlo de la siguiente forma:

temp = lectura * 500 / 1024

temp = lectura * 0.48828125

ACTIVIDAD 3

De acuerdo al ejemplo anterior, realice lo siguiente:

a) Realice el esquema de conexión correspondiente al ejemplo anterior.

b) Cablee de manera física el esquema de conexión señalado en el inciso a.

c) Haga un ajuste al programa anterior para mostrar la temperatura a través de una pantalla LCD 16 × 2.

d) Realice el esquema y el cableado físico del ejercicio propuesto.

En el desarrollo de proyectos de ingeniería habitualmente se utilizan otros sensores como sensor de gas, de humedad, de radiofrecuencia, sensores de proximidad, ultrasónicos, de distancia, etc.

Dentro de Arduino, también se puede jugar con el uso de caracteres desde el monitor serial. De este modo se ilustra la forma de enviar información desde la PC a la placa de Arduino.

6. ENUNCIADO: Escribir desde la pantalla de la PC (monitor serial) una letra predeterminada (por ejemplo la letra a). La primera vez que se escriba dicho carácter deberá encenderse un led, si se escribe por segunda vez el led se apaga. Este estará conectado en el pin n.º 13.

```
int led = 13;
char leer; //Variable donde se almacena la letra
boolean prendido=false; //Estado led la primera vez, apagado
void setup()
{
Serial.begin(9600);
pinMode(led, OUTPUT);
}
void loop()
{
    leer=Serial.read(); //Guardar en una variable el valor de la consola
serial
      if ( (leer==’a’) && (prendido==false) )
       {
          digitalWrite(led,HIGH);
          prendido=true;
}
    else if ( (leer==’a’) && (prendido==true) )
        {
           digitalWrite(led,LOW);
           prendido=false;
         }
}
```

Desde el IDE de Arduino se puede controlar muchos componentes, incluso lograr el almacenamiento sobre SD *Card*, creación de chats y registro de conversaciones (SD *Card*), hacer llamadas telefónicas (mediante el uso de módulos GSM), etc. En la presente sección de este capítulo se han mostrado algunos códigos que le permitirán al usuario poner a prueba sus conocimientos sobre diseño lógico y programación. Pues, no se trata de un libro dedicado a la plataforma Arduino.

7.4.4 Control inalámbrico

Para controlar un conjunto de componentes electrónicos (o inclusive un robot con Arduino) se han empleado muchas tecnologías. Una de las más básicas es la *shield* bluetooth HC-05 y HC-06. La *shield* bluetooth básica puede incorporar por lo regular hasta 6 pines. Dos son utilizados para polarización (VCC y GND), dos más para recepción de datos (RXD) y transmisión (TXD). Y, de manera adicional, para módulos HC-05 se tiene un pin llamado STATE y uno más llamado KEY (para activar el modo de comandos AT).

Para el diseño de proyectos que desean controlarse a distancia, originalmente se recurre a módulos HC-05 (conexión esclavo-maestro) o HC-06 (conexión en esclavo). Aunque una opción más versátil es la *shield* BTBee Pro (HC-05 compatible para XBee).

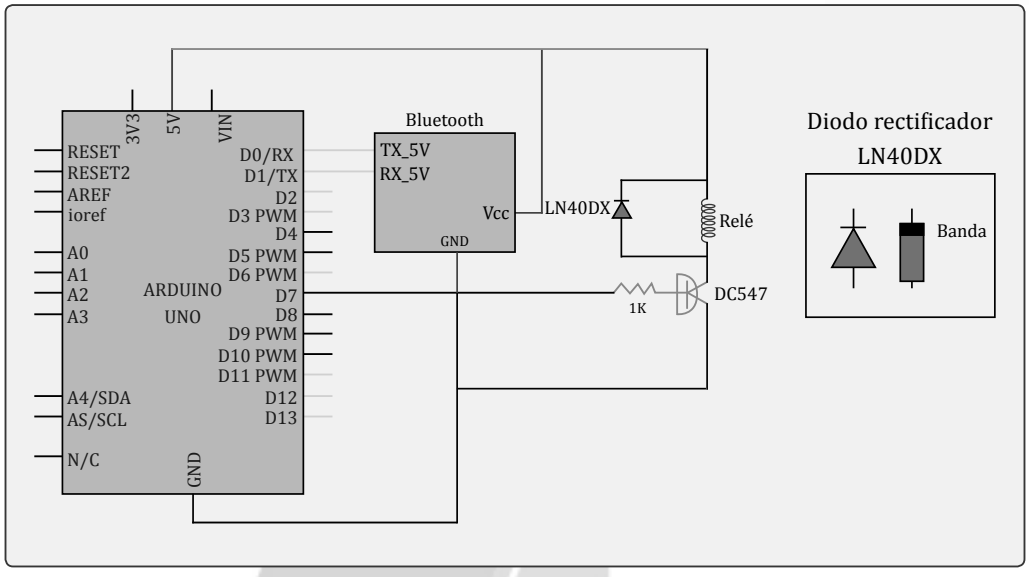


Figura 7.9 Esquema de conexión entre la placa Arduino y la *shield* bluetooth para el control de un relevador

Lo anterior es muy útil al momento de querer realizar una conexión inalámbrica (control remoto de algún hexápodo, carro, encendido de ledes o lámparas, etc). La visualización de información y operación se puede lograr de cuatro formas distintas: 1) a través del monitor serial, 2) mediante el uso de una pantalla LCD, 3) mediante alguna app.

Nota: el esquema necesario para este ejercicio se encuentra en la figura inmediata anterior.

7. **ENUNCIADO:** Controlar la activación de un relé (relevador). Se pretende utilizar el monitor serial como interfaz tanto para la visualización de datos como la operación:

```
#define rele 7
char bandera;
boolean hayDato;
void setup()
{
pinMode(rele,OUTPUT);
//iniciar comunicacion serial con velocidad de bluetooth osea 9600
Serial.begin(9600);
bandera = 0;
hayDato = 0;
}
void loop()
{
```

```
if(hayDato)
  {
  if(bandera== ‘s’ || bandera == ‘S’)
    digitalWrite(rele, HIGH);
    else  if(bandera== ‘n’ || bandera == ‘N’)
    digitalWrite(rele, LOW);
  else
  Serial.println(“COMANDO NO RECONOCIDO”);
hayDato = 0;
   }
}
void serialEvent()
{
  while(Serial.available() > 0)//este while pregunta si hay datos existen-
tes
  {
  bandera = Serial.read();
  Serial.print(“Dato recibido: “);
  Serial.println(bandera);
hayDato = 1;
    }
  }
```

7.5 *SCRATCH* (S4A)

Una de las herramientas más conocidas para la generación de algoritmos de programación es, sin duda, *Scratch*. Esta versátil herramienta ha sido la protagonista en gran cantidad de proyectos de índole informático. Fue desarrollado por el MIT (el cual mantiene su licencia) y actualmente hace su aparición en el mágico mundo de Arduino.

S4A (*Scratch* 4 Arduino) es una versión ideal para trabajar con la plataforma de hardware libre. Su interfaz es muy similar a la original y permite la construcción de algoritmos mediante módulos de programación. Funciona como IDE, a través del cual se verifica y se envía la codificación creada al microcontrolador de la placa de Arduino.

Nota de interés

Recursos S4A

El portal de descarga de *Scratch* para Arduino es: http://s4a.cat. A través de este enlace es posible la consulta de una gran variedad de información técnica, recursos, ejemplos, esquemas, proyectos, e incluso el propio *firmware* del S4A. Aunque el sitio está en inglés no deja de ser interesante y muy intuitivo.

Otro recurso son algunas fichas educativas, las cuales le muestran al usuario la forma de conectar una placa de Arduino con S4A. Estas fichas se pueden consultar a través de https://filobotica.wordpress.com/2015/04/29/scratch-4-arduino-fichas-educativas.

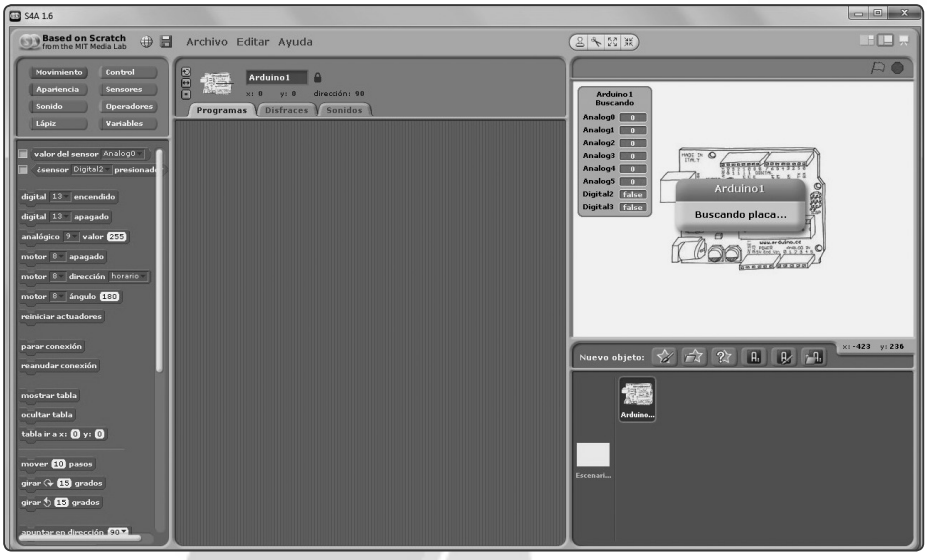


Figura 7.10 Ventana principal de *Scratch* para Arduino (S4A)

Scratch para Arduino requiere de igual modo de su instalación sobre la PC, la carga del controlador (que previamente se ha explicado) y la carga del firmware (programa que debe instalarse en la placa Arduino para mantener una comunicación desde S4A). Actualmente, la mayoría de las placas son compatibles con este

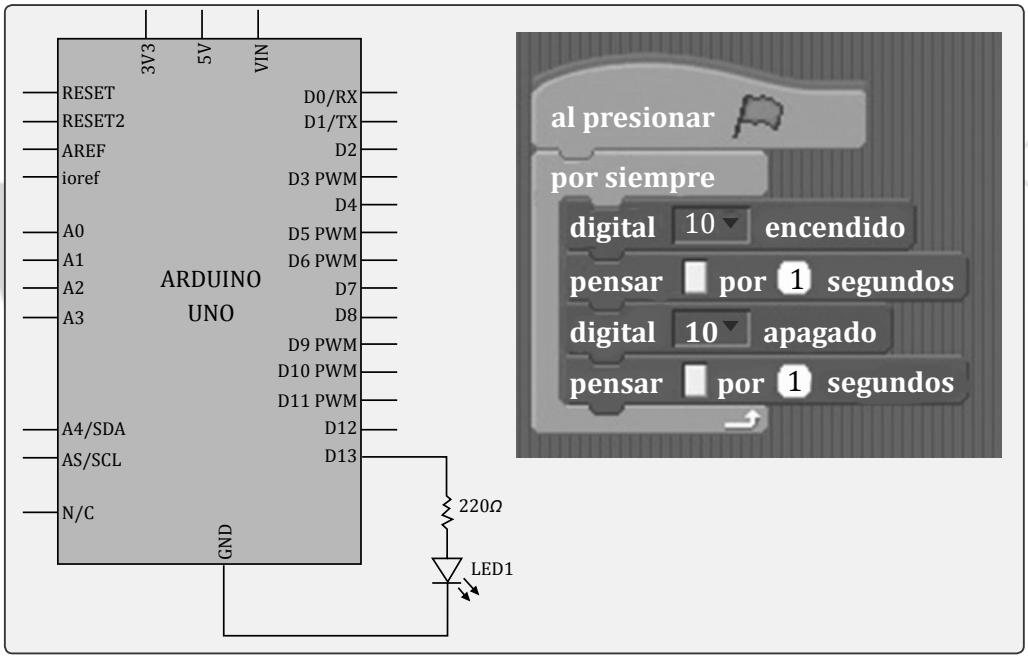


Figura 7.11 Esquema y programa en S4A. Permite el encendido y apagado de un led
Fuente: el autor.

7.6 DESARROLLO DE APP PARA MÓVILES

Una app móvil se define como aplicación de software que se instala en dispositivos móviles (celulares o tabletas electrónicas) para satisfacer la necesidad en concreto de un usuario, ya sea de índole profesional o de ocio y entretenimiento. Es importante no confundirla con una Web app que no es instalable.

Para la creación de una app, a menudo se pueden utilizar soluciones de software como AppInventor, DroidDraw y PhoneGap, las cuales cumplen con el propósito de hacer posible la integración de una aplicación a través de módulos. Sin dejar a un lado a JavaScript, HTML5 y CSS3 definidas como herramientas fundamentales para el programador de aplicaciones móviles.

Una app puede ser programada desde cualquier PC (bajo cualquier sistema operativo: Windows, GNU Linux, Mac OSX) utilizando alguna de las herramientas de software antes mencionadas.

Actualmente y gracias al avance tecnológico, es posible la integración de una app a cualquier dispositivo, independientemente del sistema con el que opere, aunque el de más fama es el sistema Android.

Android es un sistema operativo para dispositivos móviles muy popular en el mercado. Este sistema permite la posibilidad de incluir aplicaciones diseñadas por una comunidad de usuarios. Esta última tarea es relativamente sencilla, aunque para llevarla a cabo se requiere de herramientas como las antes mencionadas para la construcción de interfaces (AppInventor, es uno de los más comunes o, en su defecto, la *suite* de Android Studio). El trabajo con Android implica el manejo de paquetes APK. Esta se define como la extensión de un archivo propio del sistema Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android en teléfonos celulares y tabletas.

Las herramientas para el diseño de app, a menudo facilitan la interacción del usuario con los módulos o dispositivos de hardware, proyectos electrónicos y de robótica. Así como el control de los mismos. Para esto basta programar una aplicación y cargarla en el dispositivo móvil (sea un celular o una tableta) para ejercer la función de control de mando.

En la actualidad, desde Google Play se pueden obtener miles de app que interactúan con piezas electrónicas, proyectos, aparatos domésticos, redes, sin tener que programarlas o desarrollarlas desde cero. Y desde luego, le permiten al usuario su modificación para el ajuste a sus necesidades. Más adelante se utilizará *Bluetooth RC Controller* (aplicación de control remoto por bluetooth) y ROMPI (efectuar el control de un robot mediante tonos).[2]

[2] Consiste en una aplicación para dispositivos móviles que permite la creación, edición, compilación y ejecución de programas escritos en lenguaje LCS. Ideal para Android, aunque permite su carga en dispositivos con sistemas distintos. Fue creado por la UNAM.

Nota de interés

Creación de app móviles

En Internet abunda una gran variedad de recursos para la creación de app para móviles. Algunas opciones interesantes se ubican en los siguientes enlaces son: www.appmakr.com/es/, www.mobincube.com/es/ y www.androidcreator.com/es/.

7.7 ROBÓTICA CON ARDUINO

Arduino se ha convertido en una herramienta vital para el desarrollo de proyectos de robótica, esto es gracias a la simplificación de funciones que integra su microntrolador. Pues este, entre otras funciones, hace posible la emisión de pulsos al componente o dispositivo que desea controlarse, el cual depende, necesariamente, del propósito del robot.

Hay que recordar que un robot es un aparato o sistema eléctrico y mecánico que es capaz de realizar el trabajo que realiza un ser humano. Los robots son muy utilizados en la industria y efectúan tareas muy específicas (ensamble de componentes, soldadura, estibación de productos, desplazamientos, etc). Estos a veces deben ser programados (dotarlos de un conjunto de instrucciones) para que cumplan con un propósito en especial. Actualmente se ven funcionar en elevadores, juegos mecánicos, electrodomésticos, etc. y tienen la finalidad de auxiliar a cualquier labor cotidiana.[3]

Los robots, por lo general, están integrados por múltiples piezas y sistemas de control que hacen posible la ejecución de tareas específicas. Estos por lo regular contemplan cuatro fases (mecánica, eléctrica, electrónica, informática), según el Dr. Enrique Ruiz Velasco.[4]

Muchas máquinas industriales utilizan motores, que son controlados por placas, que permiten su desplazamiento, movimiento, iteración o rotación (por lo regular de forma automática).

Nota: Un sistema de control se define como un módulo de hardware diseñado para abastecer una necesidad en particular. La ingeniería de hardware ha hecho posible el diseño de placas controladoras de movimientos a través de la generación de pulsos.

[3] Para profundizar sobre el tema de robótica y electrónica, consúltese Ruiz, E. (2007). *Educatrónica: innovación en el aprendizaje de las ciencias y la tecnología*.

[4] Profesor de la UNAM e investigador en el Instituto de Investigaciones sobre la Universidad y la Educación (IISUE).

7.7.1 La importancia del motor en la robótica

El motor es un elemento muy comúnmente utilizado en el campo de la robótica. Este posee ciertas características y parámetros que deben ser considerados a la hora de adquirirlo. Los hay de diferentes tipos y tamaños. Algunos tipos son:

- **Motor DC:** Es el motor de corriente continua, se puede obtener de algún aparato electrónico, como consolas, reproductores de discos, impresoras, lavadoras, juguetes, etc.
- **Motorreductor:** Permite reducir su velocidad de rotación. Son utilizados en la industria y en la elaboración de juguetes de colección electrónicos de propósito específico (aviones, submarinos, carros, helicópteros, drones). A menudo son diseñados a base de engranajes, mecanismos circulares y dentados con geometrías especiales de acuerdo con su tamaño y la función en cada motor. Es capaz de soportar una masa determinada.
- **Servomotor:** Este tipo de motor permite trabajar con posiciones angulares específicas. Son usados en la robótica y mini robótica (diseño de brazos, carros, guantes, humanoides, etc.)

Algunos parámetros que se deben considerar para la adquisición de un motor y su aplicación en el desarrollo de un proyecto de robótica son:

- **Tensión o voltaje:** Está expresado en volts. Las tensiones más populares para su manejo en robótica oscilan entre 1.5 V y los 12V.
- **Corriente absorbida:** Es la cantidad de corriente (I) en mA (miliamperes) o A (amperes) que el motor requiere de la fuente de poder. A menudo se emplea la ley de Ohm para realizar cálculos.
- **Velocidad de giro:** Se expresa en revoluciones por minuto (rpm), la mayoría de motores de DC tienen una velocidad normal de operación de 4000 a 7000 rpm, para aplicaciones en la robótica estas velocidades son muy grandes, es por eso que se recurre a circuitos electrónicos para poder regular esta velocidad a valores adecuados para el manejo de brazos y pinzas en un robot.
- **Torque (par):** Se define como la fuerza que el motor ejerce sobre la carga. Al reducir el torque, el motor reducirá su potencia. Al reducirse considerablemente el motor demandará mayor potencia de la normal, lo que ocasionaría un sobrecalentamiento que podrá ocasionar un daño al motor. Es importante consultar las especificaciones o datos técnicos del fabricante (desde allí se podrán consultar fórmulas y variables de suma importancia para efectuar los cálculos respectivos).
- **Reductor de velocidad:** Esta especificación es exclusiva para motorreductores. Este tipo de motores son apropiados para el accionamiento de toda clase de máquinas y aparatos de uso industrial que necesitan reducir su velocidad en una forma segura y eficiente.

7.7.2 Control de motores

Controlar un motor desde la placa Arduino es tarea fácil, pues basta con polarizarlo para que funcione. Sin embargo, si se desea controlar la velocidad de giro por ejemplo, se requerirá de un procedimiento más laborioso como la configuración y programación de algún Pin PWM de la placa. Ahora, si lo que se requiere es hacerlo girar a una determinada dirección, debe emplease un puente H. Este último elemento es útil cuando se trabaja con motores DC que pretenden ser controlados para reducir su velocidad o hacerlo girar a una dirección específica.

Nota: Un puente H es un circuito, cuyas conexiones están dispuestas de tal forma que conforman una letra H. Este puente es muy utilizado para el manejo de un conjunto de motores que desean manipularse. Este arreglo, por lo general, está compuesto por transistores y diodos rectificadores, principalmente. Su conexión prevé el movimiento de un motor en distintas direcciones (adelante, reversa, derecha, izquierda e incluso mantenerlos en STOP).

Actualmente, es posible el desarrollo de circuitos o sistemas de control. Por ejemplo, para lograr girar un motor DC a una velocidad deseada por ciertos lapsos de tiempo, es necesario crear un circuito que permita dicha rotación. A continuación, se muestra un diagrama de conexión para construir un sistema de control básico para hacer girar un motor en diferentes direcciones, este puede ser controlado desde el microcontrolador ATMega.

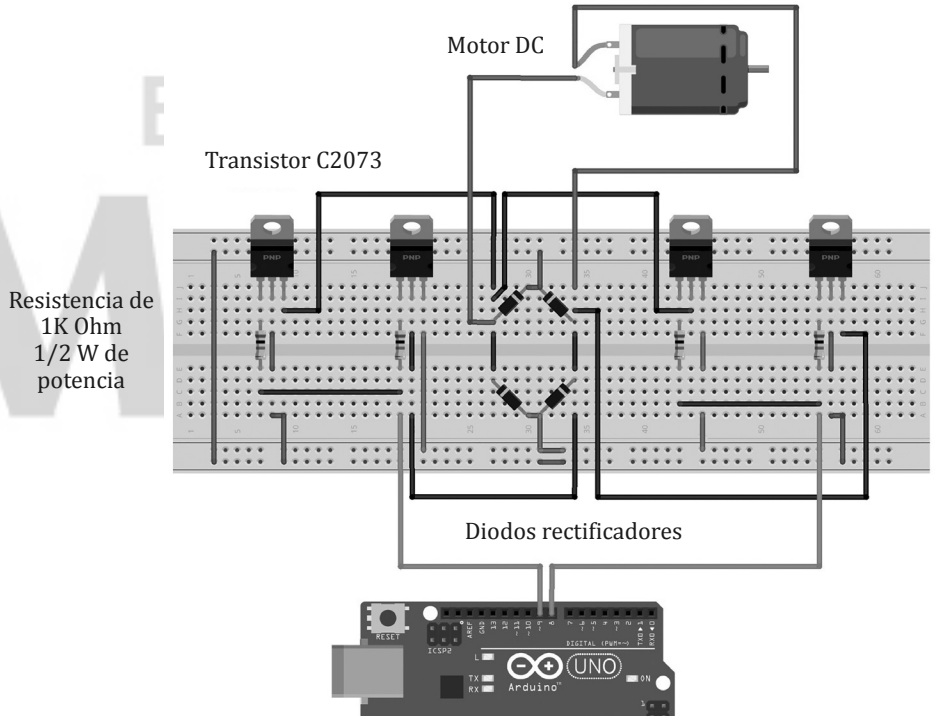


Figura 7.12 Diagrama de conexión de puente H para manipulación de un motor DC mediante la placa Arduino UNO

El diagrama anterior puede ser muy útil a la hora de generar proyectos como un carro seguidor de línea, un helicóptero controlado por celular, una banda transportadora, una cortina para garaje, etc., ya que es posible controlar su velocidad de giro, su sentido y suministro de potencia. Lo anterior, mediante el empleo de técnicas de control de motores DC. Este control puede llevarse a cabo mediante tiristores (uso de puentes H) y un conocimiento básico de electrónica de potencia (uso de potenciómetros). El *sketch* desde la IDE es el siguiente:

```
//Código para utilizar motor con puente H
int LeftPin=8; //Pines de salida del Arduino
int RightPin=9;
int input=0;

void setup(){
    Serial.begin(9600);
    pinMode(LeftPin, OUTPUT);
    pinMode(RightPin, OUTPUT);
}

void loop(){
    if (Serial.available()){
    if (input=='1'){ //El motor girará a la derecha
    digitalWrite(LeftPin, LOW);
    digitalWrite(RightPin, HIGH);
    }
    else if (input=='2'){ //El motor girará a la izquierda
    digitalWrite(LeftPin, HIGH);
    digitalWrite(RightPin, LOW);
    }
    else if (input=='0'){ //El motor se detendrá
    digitalWrite(LeftPin, LOW);
    digitalWrite(RightPin, LOW);
    }
    delay(10);
    }
}
```

Gracias a los avances de la electrónica, se ha dado paso al desarrollo de componentes electrónicos que hacen posible el control de motores. Uno de ellos es el circuito L293D y el driver L298 (con doble puente H). Ambos simplifican el trabajo y evitan así su cableado manual. Estos componentes a menudo se encuentran integrados por un puente H doble interno (que permite el control de dos a cuatro motores según corresponda).

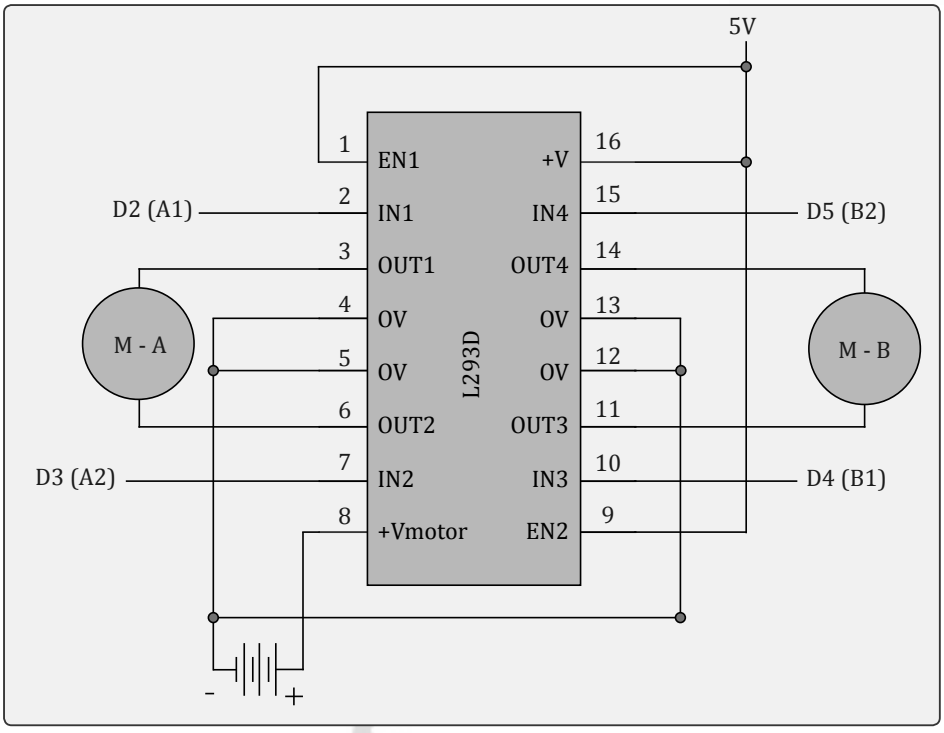


Figura 7.13 Diagrama de conexión del circuito L293D

Actualmente existen placas controladoras de motores basados en el driver L298 (como es el caso de la placa L298N) que puede controlar cuatro motores con solo dos salidas (A, B de 25 W). Los cuatro dispuestos en pares con una conexión en paralelo. Para alimentar un conjunto de cuatro motores como máximo (en el caso de fabricación de un carro seguidor de línea o control remoto) serán suficientes de 7,5 voltios.

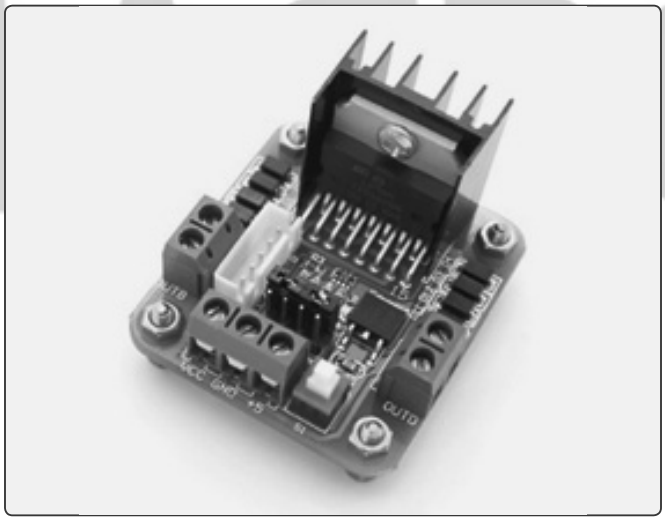


Figura 7.14 PCB L298N, controladora de dos motores DC compatible con la placa Arduino

Es importante saber que una placa Arduino no puede gestionar directamente motores de corriente continua, dado que la máxima intensidad capaz de proporcionar en sus pines de salida es de escasos 20 mA y no mayor a 50. Por lo tanto, se requiere de un controlador de motores (compatible con Arduino) que sea capaz de soportar su carga (cuya intensidad varía de 300 mA a 1000 mA). Una opción muy válida para solucionar este problema es utilizar un transistor como interfaz entre el Pin PWM y el motor.

Ejemplo:

Si un motor trabaja con una tensión de 6 V y consume 300mA a plena carga, ¿cuál es la intensidad de corriente necesaria? Para realizar este cálculo, se tiene que multiplicar el número de motores existentes en un circuito por la intensidad (I). Proporcionando así una cantidad exacta en Amperios. Por ejemplo, 4 motores x 300 mA = 1,2 amperios aproximadamente.

Actualmente, muchos robots utilizan a menudo lo que se conoce como motores de propósito específico (el motorreductor y el servomotor). En el siguiente tema de este capítulo se comenzará a hablar sobre el manejo de un servomotor desde la plataforma Arduino.

A. El servomotor

Los servomotores son elementos que poseen un eje de rendimiento controlado, este puede ser llevado a posiciones angulares específicas al enviar una señal codificada. En la práctica se utilizan para posicionar superficies de control como el movimiento de palancas, ascensores, brazos electrónicos, juguetes y con frecuencia son utilizados en mecanismos de seguridad como cerraduras. Trabajan con grados de precisión. En robótica son muy usados en creación de hexápodos (arañas electrónicas) o gusanos.

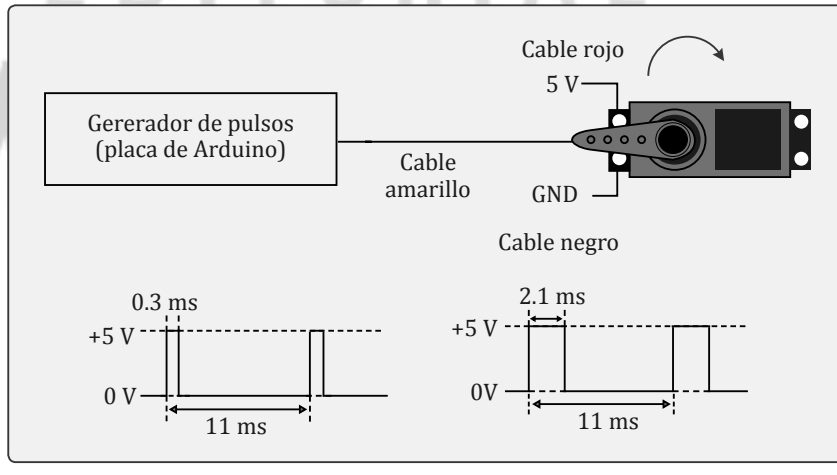


Figura 7.15 Distribución de cables de un servomotor y esquema de ancho de pulsos

Un servomotor de posición está conformado por diversos elementos internos, entre los que se encuentran un motor DC, un potenciómetro, un conjunto de engranes y un circuito de control o PCB (codificador). Contiene tres cables: un rojo (+5V), un negro (GND) y un cable amarillo para conectarlo a un generador de pulsos. Estos elementos trabajan en conjunto para ofrecer el desplazamiento de un pequeño brazo actuador hasta 180° (los cuales pueden ser incluso configurados para girar hasta 360°). Para determinar los ángulos de desplazamiento de dicho motor, este debe programarse.

Usualmente, este tipo de motores hace uso de baterías especiales LiPo para evitar problemas de comportamiento al momento de suministrar voltaje. Se recomienda el uso de baterías de Litio de 1000 mA y 3.7 V.

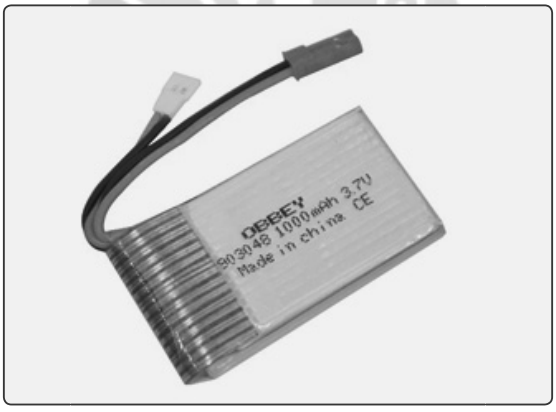


Figura 7.16 Batería LiPo, ideal para trabajar con servomotores y pantallas LCD 16 × 2

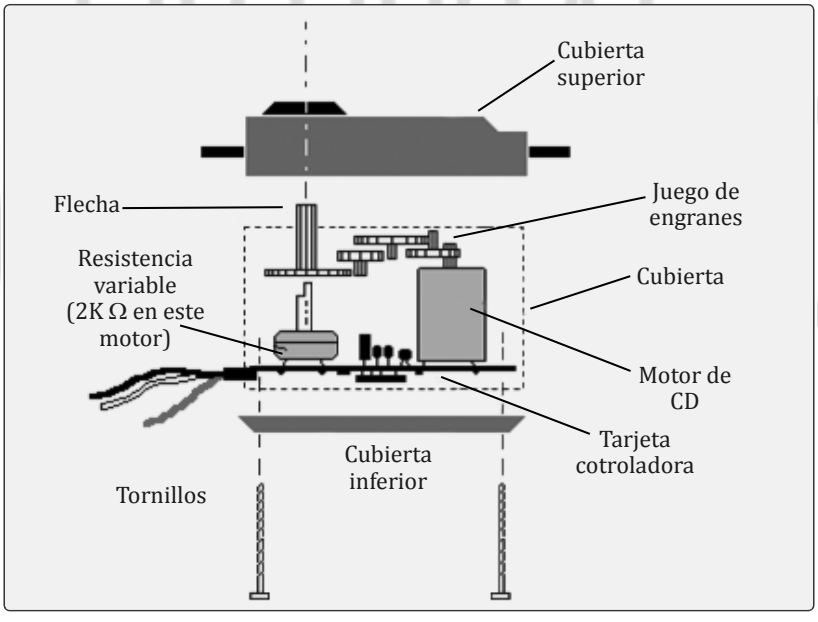


Figura 7.17 Esquema de las partes internas de un servomotor de posición

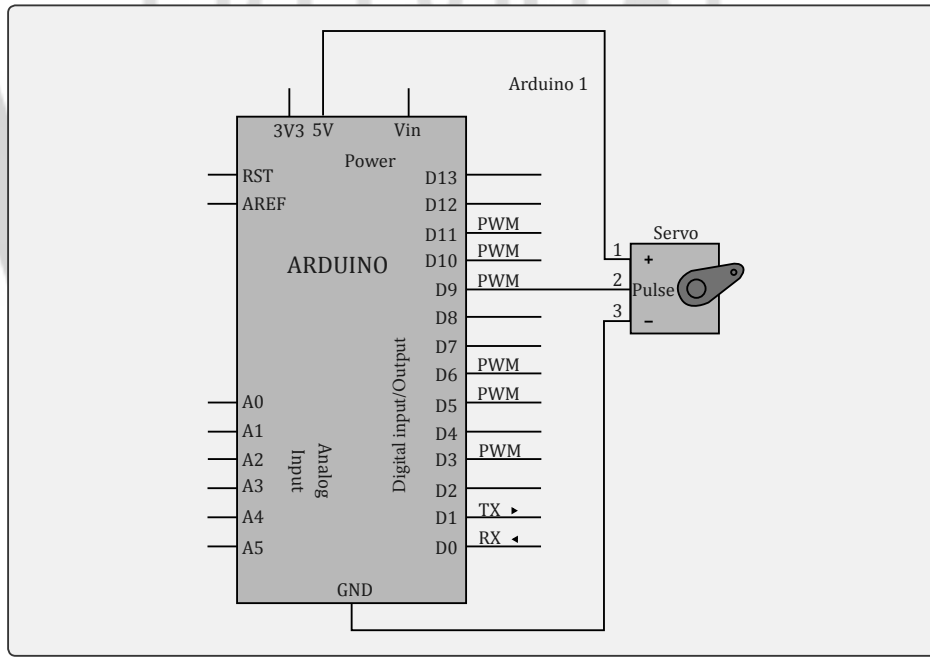
Ejemplo de codificación de *sketch* en Arduino

ENUNCIADO: Conseguir una rotación del servomotor a 180° por determinado tiempo, y de regreso. Se considera el Pin 9 para la conexión del Servo.

Sketch

```
/* Sweep
#include <Servo.h> /librería para manejo de Servo

Servo myservo;  // creación del objeto
int pos = 0;    // variable posición del servomotor

void setup()
{
  myservo.attach(9);  // Pin9 para el objeto creado
}

void loop()
{
  for (pos = 0; pos <= 180; pos += 1) // grados
     {
        myservo.write(pos);
       delay(15);
     }
for (pos = 180; pos >= 0; pos -= 1)
      {      myservo.write(pos);  delay(15);
      }
}
```

Conexión



Nota de interés

Lego Mindstorms

Lego ofrece la oportunidad de programar y construir un robot a medida. Para ver más información sobre este tipo de prácticas, puede consultar la siguiente página: http://www.lego.com/es-ar/mindstorms/support. Desde allí, puede también obtener información respecto a servomotores interactivos y programación en NXT Programming.

B. Motorreductor

Los motores que poseen la capacidad de reducir su velocidad (aunque internamente la fuerza del motor integrado aumenta su fuerza), originalmente utilizados en la industria (por ejemplo para máquinas de operación o el desarrollo de sistemas de entretenimiento como juguetes de control remoto, también en aparatos como reproductores de CD, DVD, etc), se llaman motores de reducción. Los hay de distintos tipos, tamaños, pesos, relación de reducción (**X:XX**) y fabricantes. A menudo integran un torque (par) máximo a la salida, expresado en Kg/m. Para su ejecución sobre un equipo trabajan con ejes de 180° y 90°, con un eje de salida vertical u horizontal.

Para proyectos de robótica y electrónica se venden kits completos parar armar robots. De hecho, en la actualidad es posible encontrar motorreductores con llanta para integrarlos a un coche de control remoto. Aunque siempre es posible desarrollarlo manualmente desde cero (aplicando la filosofía DIY o *Do It Yourself*).

Figura 7.18 Motorreductor con llanta. Ideal para carritos seguidores de línea

7.7.3 Proyecto de robótica con Arduino

En esta sección se propone la construcción de un robot diferencial (carro controlado desde un dispositivo móvil). Para ello se utilizará algunos de los elementos antes mencionados, una app para Android y, desde luego, algunas funciones de lenguaje C empleadas en el entorno de programación de la placa Arduino. Para esta tarea, en primera instancia, se listan los materiales necesarios:

- Placa Arduino UNO.
- Dos motores DC bipolares.
- Un circuito L293N (puente H).
- *Shield* Bluetooth HC-05.
- Cables (de preferencia con Jumper integrado).
- Protoboard.
- Fuente de alimentación (adaptador de AC a DC universal, con salidas de 1.5V - 12V).
- Soporte o chasis para carro.
- Dos llantas de goma para los motores y una llanta loca.

-App **Bluetooth RC Controller**

Se muestra el esquema de conexiones generales, en el que se puede apreciar cinco elementos primordiales: la placa Arduino UNO, el circuito L293D, la *shield* bluetooth, los dos motores y la fuente de alimentación o batería, cuyo voltaje estará en función de las características del tipos de motor empleado.

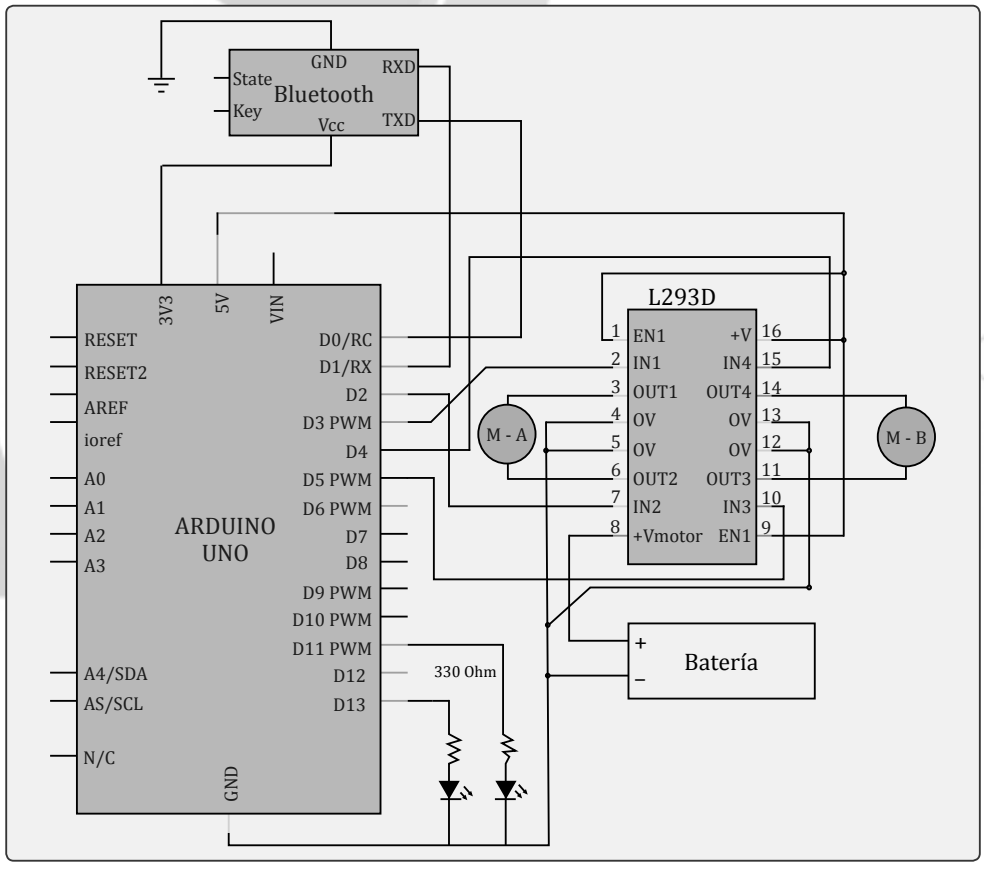


Figura 7.19 Esquema de conexión para armar un vehículo controlado desde un dispositivo móvil

Ahora, de la programación de este proyecto. Esta fase se divide en dos:

1. **Programación de la placa Arduino:** Consiste en el código fuente o *sketch* que se debe capturar desde el IDE de Arduino.
2. **Programación de la interfaz del control de mando:** Consiste en diseñar un programa (app) que será cargado en el dispositivo móvil empleado. En este caso, y solo con fines prácticos, bastará con descargar la aplicación disponible para Android llamada Bluetooth RC Controller desde Google Play. Su manejo es muy intuitivo, pues a través del empleo de caracteres es posible el envío de comandos a la placa Arduino, la cual interpretará para realizar las funciones necesarias.

Tabla 7.2 *Caracteres asociados para el manejo de la app Bluetooth RC Controller*

Adelante / Forward = F	Reversa / Backward = B	Izquierda / Left = L
Derecha / Right = R	A-I / Forward Left = G	A-D / Forward Right = I
R-I / Backward Left = H	R-D / Backward Right = J	Parar / Stop = S

Otra opción interesante muy parecida es Arduino Control Car, disponible desde Google Play para su descarga. A continuación, se muestra el *sketch* desarrollado desde el entorno de desarrollo de Arduino. Es importante tener en cuenta que los caracteres anteriores están declarados en el código para su manejo desde el monitor serial y la app Bluetooth RC.

```
//Identificación de ambos motores (rueda A y rueda B)
int motorA1 = 2;
int motorA2 = 3;
int motorR1 = 4;
int motorR2 = 5;
int luz = 11;
int led=13;
char c;

void adelante(); //Declaración de funciones de dirección o sentido de las
ruedas
void reversa();
void parado();
void derecha();
void izquierda();
void encenderLuz();
void setup(){
  Serial.begin(9600); // Entrada y salida de datos desde el monitor serial
  pinMode(led, OUTPUT);
  pinMode(motorA1, OUTPUT);
  pinMode(motorA2, OUTPUT);
  pinMode(motorR1, OUTPUT);
  pinMode(motorR2, OUTPUT);
  pinMode(luz, OUTPUT);
}
void loop(){
while (Serial.available()) {
    if (Serial.available() >0) {
 c = Serial.read();
     }
  }
```

```
   switch(c){ //Declaración de casos (sentido de las ruedas)
      case 'F':
        adelante();
        break;
      case 'B':
        reversa();
        break;
      case 'S':
        parado();
        break;
      case 'L':
        izquierda();
        break;
      case 'R':
        derecha();
        break;
    }
  }

void adelante(){ // Funciones de dirección o sentido de las ruedas
  digitalWrite(motorA1,HIGH);
  digitalWrite(motorA2,HIGH);
  digitalWrite(led,HIGH);
}

void reversa(){
  digitalWrite(motorR2,HIGH);
  digitalWrite(motorR1,HIGH);
  digitalWrite(led,HIGH);
}

void parado(){
  digitalWrite(motorR2,LOW);
  digitalWrite(motorR1,LOW);
  digitalWrite(motorA2,LOW);
  digitalWrite(motorA1,LOW);
  digitalWrite(led,LOW);
}

void parado2(){
  digitalWrite(motorR2,LOW);
  digitalWrite(motorR1,LOW);
  digitalWrite(motorA2,LOW);
  digitalWrite(motorA1,LOW);
  digitalWrite(led,LOW);
}

void izquierda(){
  digitalWrite(motorA1,HIGH);
  digitalWrite(motorR2,HIGH);
  digitalWrite(led,HIGH);
}

void derecha(){
  digitalWrite(motorA2,HIGH);
  digitalWrite(motorR1,HIGH);
  digitalWrite(led,HIGH);
}

void encenderLuz(){ //Funciones adicionales para led
  digitalWrite(luz,HIGH);
}
void apagarLuz(){
  digitalWrite(luz,LOW);
}
```

Al oprimir las teclas correspondientes a los caracteres antes señalados en el *sketch* desde el monitor serial, el sentido de ambas llantas deberá cambiar según corresponda. Ahora, si se desea controlarlos, a través de un dispositivo móvil, basta con cargar la app Bluetooth RC Controller y manipular la interfaz.

Si los pines de RXD y TXD se han conectado de manera correcta tanto en la placa Arduino como en la *shield* bluetooth, la comunicación deberá establecerse sin problema alguno.

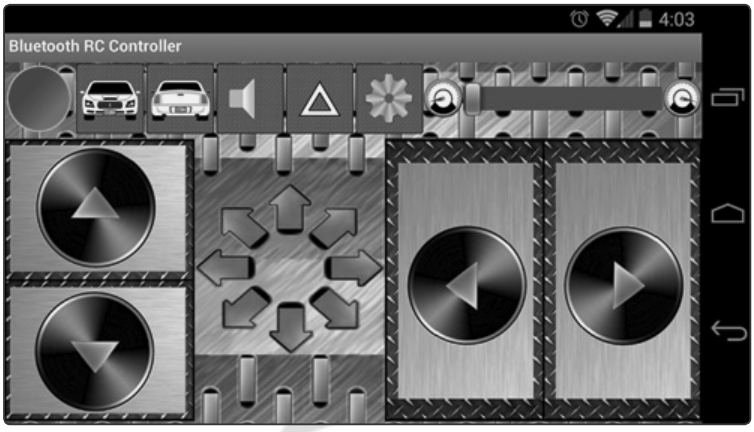


Figura 7.20 Interfaz gráfica de la app Bluetooth RC Controller

Una vez concluido dicho proyecto, se puede agregar un sensor ultrasónico, el cual hace posible medir la distancia entre el vehículo y el obstáculo que encuentre en frente (o en otra posición, dependiendo de la colocación del sensor) y así evitar un choque. Opcionalmente, está la posibilidad de incluir una pantalla LCD 16 × 2 para mostrar un conteo de obstáculos al ser detectados.

El circuito de este proyecto puede desarrollarse sobre alguna placa de circuito impreso o PCB, la cual le daría mejor vista, además de simplificar el cableado y evitar que los componentes se salgan de su lugar dentro del protoboard.

Otros proyectos adicionales que se pueden realizar, con base en la idea anterior, pueden ser carros seguidores de línea, o seguidores de luz, gusanos, hexápodos, una rueda de la fortuna, etc.

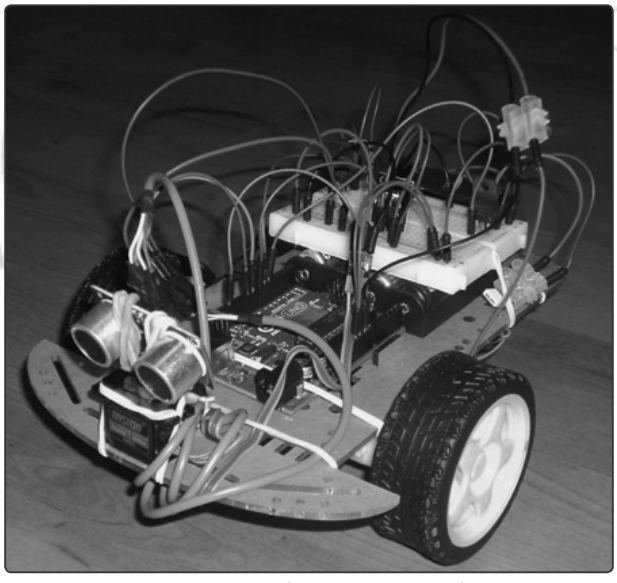


Figura 7.21 Proyecto terminado. Carrito a control remoto, controlado con un dispositivo móvil

ACTIVIDAD 4

1. ¿Cuál es el nombre del microcontrolador de la placa Arduino UNO?
2. ¿Cuáles son las estructuras de control más comunes utilizados en lenguaje C?
3. ¿Cuál es el procedimiento para preparar una placa Arduino para su funcionamiento?
4. ¿Para qué sirven la función void setup() y void loop() del IDE de Arduino?
5. ¿Qué función tiene *digitalWrite* y *digitalRead* dentro de la programación en Arduino?
6. Mencione por lo menos el nombre de tres sensores utilizados en Arduino.
7. ¿Cuál es la diferencia entre una *shield* bluetooth HC-05 y HC-06?
8. ¿Para qué sirve la herramienta *Scratch*?
9. Mencione el nombre de tres herramientas de software, como mínimo, para crear app para Android.
10. ¿Cuál es la diferencia entre un motor DC, un motorreductor y un servomotor?

Práctica de laboratorio n.º 8

Manejo de Arduino
Ledes y pulsadores

Objetivo general: Conocer la placa Arduino, la forma de montar un circuito electrónico para después programarlo y, finalmente, ejecutarlo.

Objetivo específico: Aprender a emplear pulsadores para controlar el encendido y apagado de un led y hacer variar su intensidad de luz.

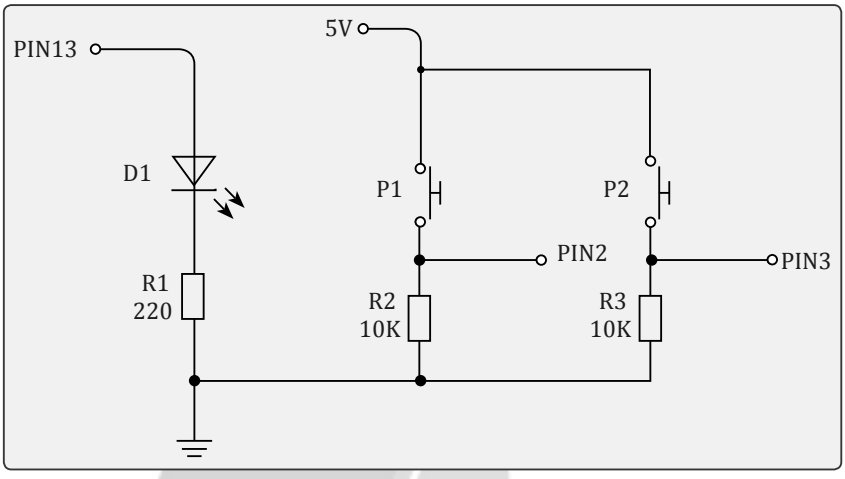
Materiales:

- Tres pulsadores.
- Un led de cualquier color.
- Dos resistencias de 10K ohmios.
- Una resistencia e 220 ohmios.
- Una fuente de alimentación de 5 V.

Procedimiento:

Para conectar un pulsador se utilizará un divisor de tensión, tal y como se aprecia en la figura de abajo, con una resistencia en *Pull-down*, se conseguirá que, al pulsar el botón, la entrada digital tome el valor de un nivel de tensión alto (HIGH).

En primer lugar debe declararse el pin como entrada y usar la función *digitalRead*() para leer el valor de la misma.



Dado el esquema de conexión anterior, monte los siguientes ejercicios:

a. Al presionar el pulsador el led se enciende. Al soltarlo, debe apagarse. El *sketch* es el siguiente:

```
int buttonPin = 2;
int ledPin = 13;
int estado = 0;
void setup()
 {
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
 }
void loop()
{
   estado = digitalRead(buttonPin);
   if (estado == HIGH)
    {
      digitalWrite(ledPin, HIGH);
    }
else {
digitalWrite(ledPin, LOW);
        }
    }
```

b. Con el uso de dos pulsadores (p1 y p2), al presionar el pulsador p1 se enciende el led, pero al accionar el pulsador p2 debe apagarse.

```
int ledPin = 13;
int inputPin1 = 2; // pulsador 1
int inputPin2 = 3; // pulsador 2
int p1;
int p2;
void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(inputPin1, INPUT);
  pinMode(inputPin2, INPUT);
}
void loop()
{
  p1=digitalRead(inputPin1);
  p2=digitalRead(inputPin2);
      if (p1 == HIGH)
       {
        digitalWrite(ledPin, HIGH);
       }
      else  if (p2 == HIGH)
   {
      digitalWrite(ledPin, LOW);
    }
}
```

c. Con el uso de dos pulsadores se pretende variar la luminosidad o brillo que emite el led. Para aumentar el brillo, se pulsa p1; para atenuar, p2. La conexión del led se conecta al Pin 9, que usa la salida PWM (modulación por ancho de pulso) para emular una salida analógica.

```
int ledPin = 9;
int inputPin1 = 2; // pulsador 1
int inputPin2 = 3; // pulsador 2
int p1;
int p2;
int value = 0;
void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(inputPin1, INPUT);
  pinMode(inputPin2, INPUT);
}
void loop()
{   p1=digitalRead(inputPin1);
    p2=digitalRead(inputPin2);
    if (p1 == HIGH)
    { value--; }
else  if (p2 == HIGH)
   { value++;
   }
value = constrain(value, 0, 255);
analogWrite(ledPin, value);
delay(10);
   }
```

Práctica de laboratorio n.º 9

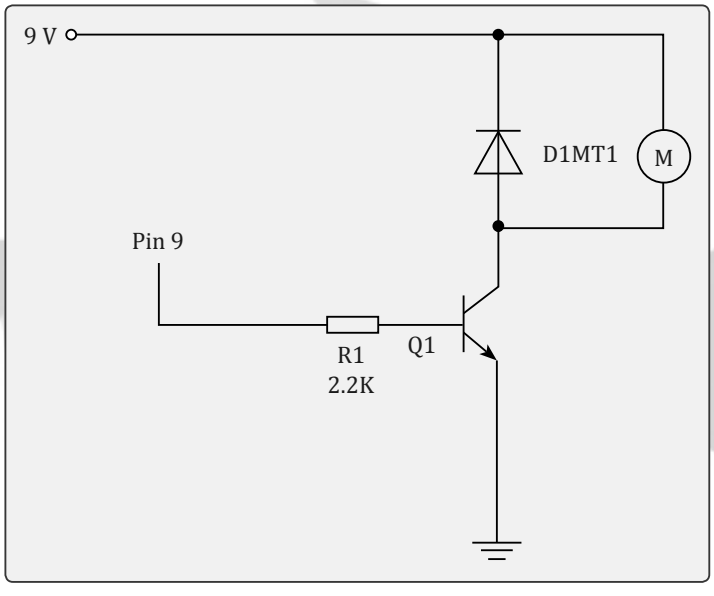
Manejo de Arduino
Control de motores

Objetivo: Conocer la placa Arduino, la forma de montar un circuito electrónico para después programarlo, y finalmente, ejecutarlo.

Objetivo específico: Conocer la forma de conectar un motor y controlar tanto su velocidad, como su sentido de giro.

Materiales:

- Un motor 5-9 V.
- Una resistencia de 2.2 K ohmios.
- Un diodo 1N4001.
- Un transistor TIP120.
- Una fuente de alimentación.



Procedimiento:

Para ilustrar mejor el contenido sobre la manipulación de un motor, se propone realizar los siguientes ejercicios:

a. Consiga la activación de un motor. Después de un tiempo, apagarlo. En el esquema anterior se puede apreciar la conexión hacia el Pin 9 de la placa Arduino con lo que conseguirá modular el pulso.

```
int motorPin = 9;

void setup()
{
   pinMode(motorPin, OUTPUT);
}
 void loop()
 {
   int onTime = 2500;
   int offTime = 1000;
   digitalWrite(motorPin, HIGH);
   delay (onTime); digitalWrite(motorPin, LOW);
   delay (offTime);
}
```

b. Se pretende controlar la velocidad del motor como si se tratara de un led al que se desea aumentar o atenuar su luminosidad. El control del motor se efectúa con la función analogWrite (pin, valor).

```
int motorPin = 9;
void setup()
 {
pinMode(motorPin, OUTPUT);
}
void loop()
{
  int onSpeed = 200;
  int onTime = 2500;
  int offSpeed = 50; // declaración de un numero entre 0 y 255 (velocidad)
  int offTime = 1000; // tiempo en milisegundos para que el motor se apague
  analogWrite(motorPin, onSpeed); // encender el motor
  delay(onTime); // espera en milisegundos
  analogWrite(motorPin, offSpeed); // apagar el motor
  delay(offTime); // espera en milisegundos
}
```

c. Consiga la aceleración y desaceleración del motor, mediante el uso de un bucle. Al igual que el ejercicio anterior, se empleará la función analogWrite().

```
int motorPin = 9;
 void setup()
{
   pinMode(motorPin, OUTPUT);
}
void loop()
{
  int delayTime = 50;

for(int i = 0; i <256; i++)
   { //aceleramos
     analogWrite(motorPin, i);
    delay(delayTime);
   }
 for(int i = 255; i>= 0; i--)
  { //frenamos
   analogWrite(motorPin, i);
   delay(delayTime);
  }
}
```

d. Haga variar la velocidad del motor usando un potenciómetro de 10K ohmios. Este último elemento debe polarizarse y después conectar su pin de control al pin 0 de las entradas analógicas de la placa Arduino.

Arduino dispone de 6 entradas analógicas y tienen un voltaje de 0 a 5 voltios, que convertidas a señales digitales se tienen valores del 0 a 1024. Esto es 10 bits de resolución.

La función analogRead(), devuelve un valor comprendido entre 0 y 1024 (10 bits) y la función analogWrite() toma valores comprendidos entre 0 y 255 (8 bits).

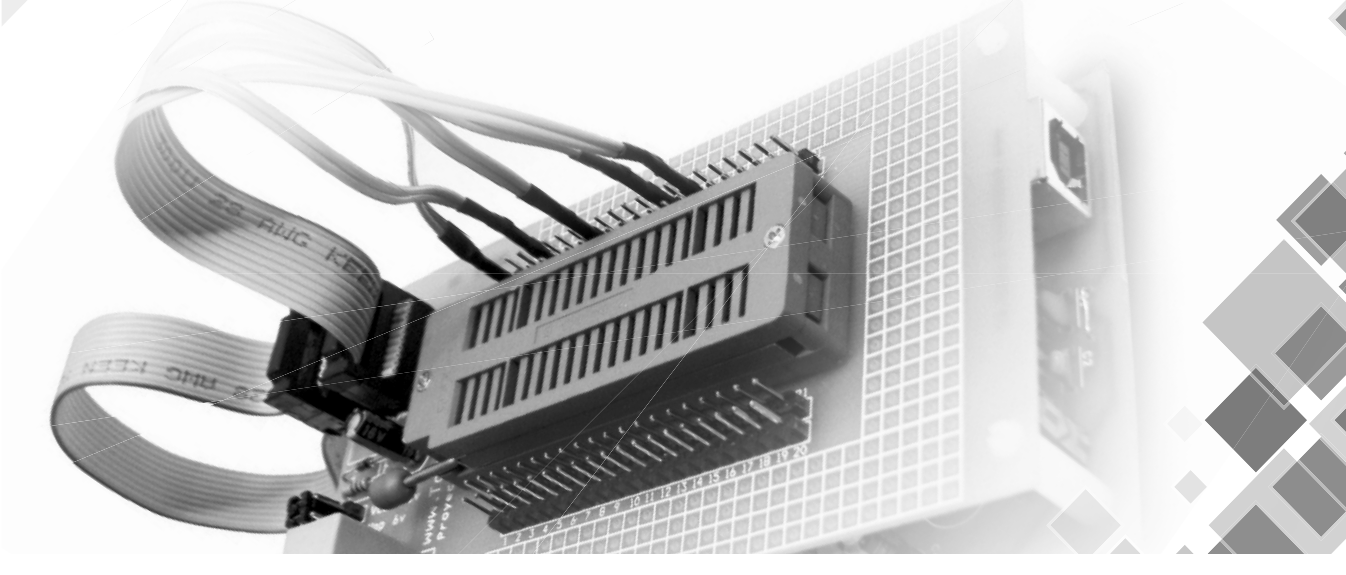
```
int motorPin = 9;
int potPin=0; // pin de control del potenciómetro al Pin 0 de la placa
int potValue;

void setup()
{
  pinMode(motorPin, OUTPUT);
}
void loop()
{
  potValue = analogRead(potPin) / 4; // al dividir 1024/4 = 255
  analogWrite(motorPin, potValue);
}
```

En este capítulo se ha hecho un estudio sobre la plataforma Arduino, su estructura, los productos que ofrece y la versatilidad que tiene al permitir la posibilidad de planear, diseñar e impulsar proyectos de índole sobre electrónica, mecatrónica y robótica. Se exponen también un conjunto de *sketch* y prácticas que le pueden servir al usuario como guía, para comenzar a incursionar con el mágico mundo de la ingeniería de hardware. En el siguiente capítulo se abordará el tema de los microcontroladores, donde se darán a conocer aspectos de gran relevancia y se desarrollarán algunos ejercicios prácticos.

EDITORIAL
MACRO®

Capítulo 8

Microcontroladores



EDITORIAL
MACRO®

Con anterioridad se ha definido el término microcontrolador, e incluso se ha descrito la diferencia que tiene con un microprocesador. Asimismo, se han mencionado algunas características, partes y funciones principales. Este capítulo queda enfocado a ilustrar el manejo de un microcontrolador, su elección, preparación, programación básica y modo de control. Para esta tarea, se ha propuesto el empleo de FlowCode como herramienta de montaje y simulación.

Es importante recordar que, para entender el mundo de los microcontroladores, a menudo se requiere de un estudio previo de unidades de almacenamiento y transferencia, sistemas de numeración, conversiones de diferentes bases, sistemas digitales, y todo lo que implica electrónica.

La situación actual en el campo de los microcontroladores se ha producido gracias al desarrollo de la tecnología de fabricación de los circuitos integrados. Este desarrollo ha permitido construir las centenas de miles de transistores en un chip. Esto fue una condición previa para la fabricación de un microprocesador. Las primeras microcomputadoras se fabricaron al añadirles periféricos externos como memoria, líneas de entrada/salida, temporizadores u otros. El incremento posterior de la densidad de integración permitió crear un circuito integrado que contenía tanto al procesador como periféricos. Así es cómo fue desarrollada la primera microcomputadora en un solo chip, denominada más tarde microcontrolador.

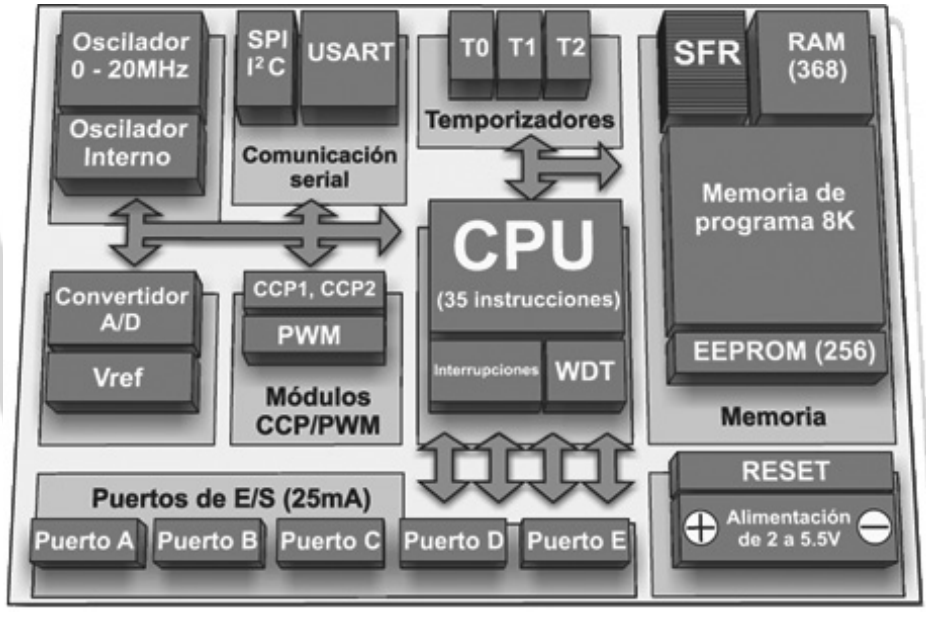


Figura 8.1 Esquema que ilustra la composición de un microcontrolador

8.1 CAMPOS DE APLICACIÓN DE UN MICROCONTROLADOR

El nombre por convención de un microcontrolador es PICmicro o controlador de interfaz periférico (*peripheral interface controller*), conocido bajo el nombre PIC. Los microcontroladores ofrecen la posibilidad de controlar prácticamente cualquier dispositivo que tenga a bien interactuar con el entorno. Es utilizado en proyectos a mediana y gran escala (industria), son muy poderosos y se encuentran disponibles en un solo chip. Los microcontroladores ofrecen una amplia gama de aplicaciones y solo algunas se exploran normalmente.

Son comúnmente empleados en la industria para el control de máquinas de distinta índole (por ejemplo, para procesos de producción/manufactura), ya que son capaces de controlar brazos mecánicos para ensamble, moldeo de piezas, estibación de cajas o andamios, etc.

Son usados en edificios inteligentes para controlar o automatizar ciertos procesos. A menudo pueden controlar sensores de luz, movimiento, lluvia, humo, fuego, etc. Controlan accesos a distintas áreas, activación de alarmas ante posibles siniestros, control de ascensores, puertas, etc.

Utilizados también en la fabricación de computadoras, dispositivos periféricos, celulares, etc.

Son utilizados a menudo en proyectos de mecatrónica, robótica, electrónica. Representa una opción alterna al uso de la placa Arduino.

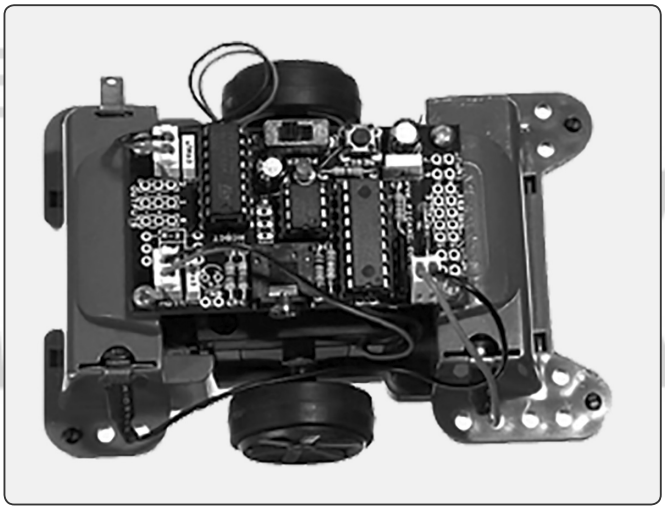


Figura 8.2 Proyecto de robótica controlado por un microcontrolador

El microcontrolador, una vez funcionando, es capaz de efectuar todas las operaciones programadas por el usuario. Otras aplicaciones pueden ser sistemas de riego, iluminación, nanotecnología, máquinas dispensadoras (golosinas, sodas, galletas, medicina, etc.), juegos mecánicos, sistemas de seguridad, automóviles, entre otros.

Figura 8.3 Los robots industriales, a menudo controlados por microcontroladores

8.2 ENTORNO, LENGUAJE Y LAS HERRAMIENTAS PARA LA PROGRAMACIÓN DE PIC

Para programar un PIC es necesario contar con un editor o entorno de desarrollo, generalmente proporcionado por el fabricante (como es el caso de los microcontroladores PICAXE y su editor PICAXE Programming Editor, o para PIC ATMEL y su editor Processing definido como un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, con el cual se ha desarrollado el IDE de Arduino).

Dos de los lenguajes de programación más utilizados en el ámbito de los microcontroladores son el lenguaje C y lenguaje ensamblador o ASM (sin dejar de mencionar algunos otros como BASIC y MikroC).

Actualmente, muchas de las funciones de un microcontrolador pueden crearse mediante el uso de herramientas de software comerciales. La más común es FlowCode para la construcción de diagramas de flujo. Otros entornos o editores para la programación de PIC son Niple, ColdFire, MPLAB, eZsystem. Algunos fabricantes de microcontroladores son Microchip Technology Inc (dueño de MPLAB CODECONFIGURATION), Dallas Semiconductors, AMTEL, Hitachi y Texas Instruments.

Nota: Es importante no confundir lenguaje de programación con entornos de desarrollo, ni tampoco confundir lo anterior con herramientas de diseño de circuitos electrónicos y simulación (como es el caso de Proteus, LiveWire-PCB Wizard, LabView, Multisim).

```
2. Counting using loops.asm: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
        RETLW 0x80
        RETLW 0x98
; } __rom_get function end

        ORG 0x0000004E
delay_s_00000
; { delay_s ; function begin
label4
        MOVLW 0xFA
        MOVWF delay_ms_00000_arg_del
        CALL delay_ms_00000
        MOVLW 0xFA
        MOVWF delay_ms_00000_arg_del
        CALL delay_ms_00000
        MOVLW 0xFA
        MOVWF delay_ms_00000_arg_del
        CALL delay_ms_00000
        MOVLW 0xFA
```

Figura 8.4 Fragmento de código de lenguaje ensamblador utilizado para programar microcontroladores
Fuente: el autor.

8.3 FLOWCODE

Es un lenguaje gráfico muy utilizado en el ámbito de los microcontroladores que no solo permite el diseño de diagramas de flujo para la generación de código en lenguaje C y lenguaje ensamblado (ASM), sino también la compilación y carga del código al PIC deseado. Cuenta con una amplia variedad de opciones, entre las cuales destacan la visualización de esquemas de distribución de pines de cientos de microcontroladores comerciales. Para mayor información sobre FlowCode, se recomienda consultar el siguiente enlace: http://www.redeweb.com/_txt/686/p50.pdf.

FlowCode es una herramienta muy útil que requiere del mínimo conocimiento de programación, pero que en ocasiones sí solicita conocimientos básicos de algoritmia. La interfaz de este software es muy amigable e intuitiva. A continuación se muestra la ventana principal de este entorno y sus componentes:

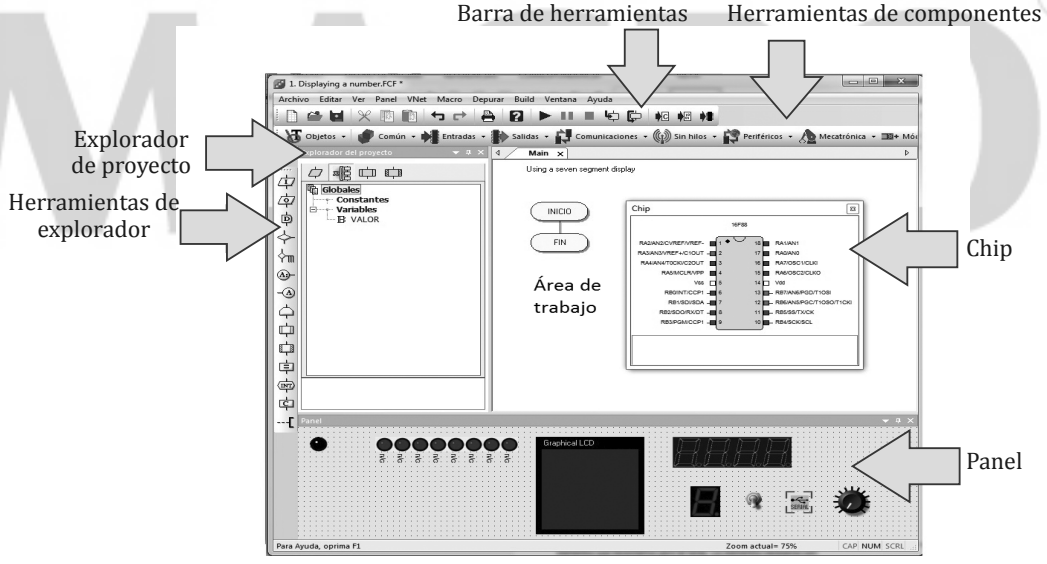


Figura 8.5 Ventana principal del entorno de FlowCode
Fuente: el autor.

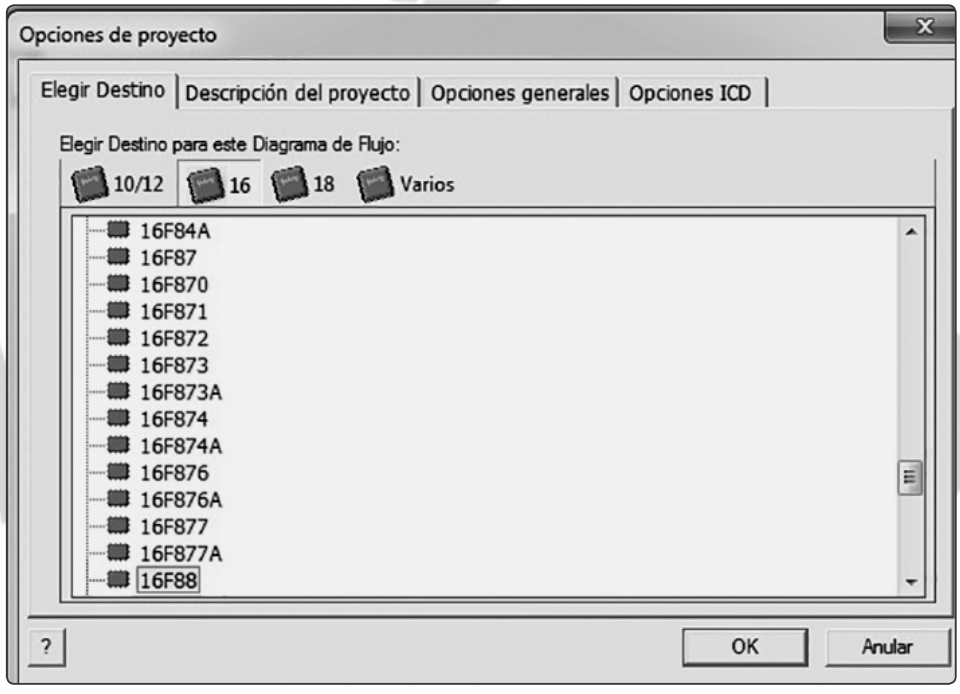
Como se puede apreciar, este entono cuenta con una gran variedad de herramientas que facilitan la construcción de diagramas de flujo, la identificación de componentes, distribución de pines o puertos de un microcontrolador, la conexión de dichos componentes electrónicos, el flujo de datos y los tipos de compilación.

Con fines didácticos, y para comenzar a familiarizarse con el software, se recomienda atender el ejemplo propuesto abajo, el cual ilustra la construcción básica de un diagrama de flujo para mostrar un valor numérico (el que desee el usuario) en un *display* de siete segmentos.

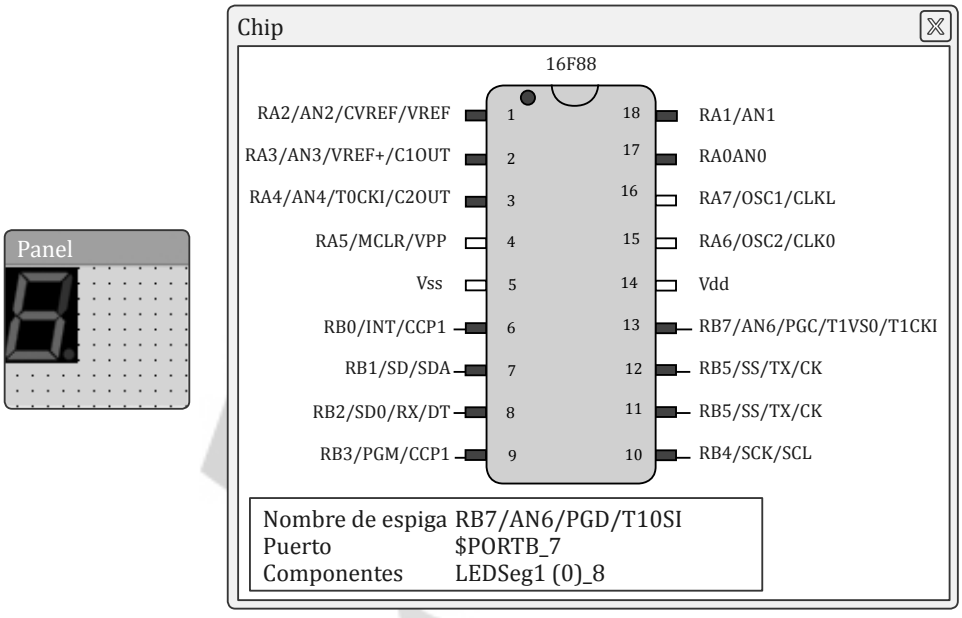
Ejemplo de uso de FlowCode

Programa: Mostrar un valor numérico cualquiera del 0 al 9 en un *display* de siete segmentos. Este valor puede ser una variable o constante definida por el usuario. El microcontrolador o chip utilizado es 16F88.

1. Al abrir la interfaz principal, se procede a elegir el chip deseado. Para ello, se debe dar clic sobre la barra de menú en la opción **Build** > **Opciones de proyecto** y desde la pestaña **Elegir destino** se debe seleccionar el 16F88. Si por alguna razón, no se ve el esquema correspondiente, debe dar clic sobre barra de menú en la opción **Ver** > **Chip**, y actualice su búsqueda.

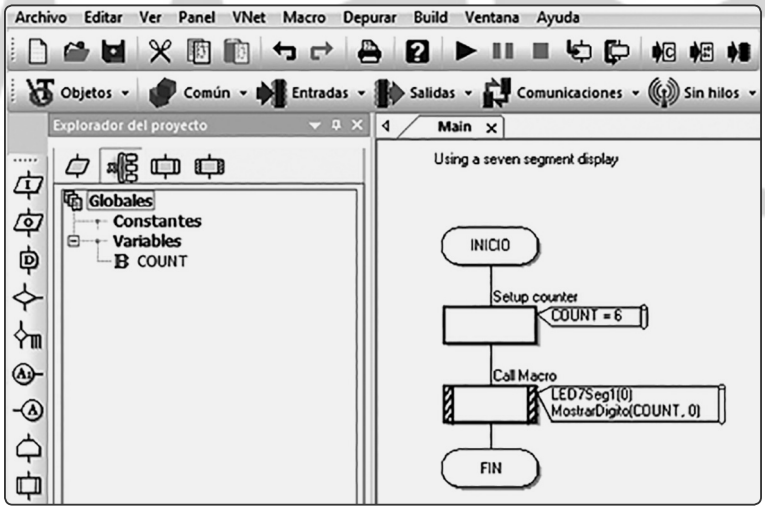


2. Una vez insertado el chip, se deberá analizar los pines que se van a necesitar para su configuración con el *display* de siete segmentos, el cual se inserta oprimiendo clic en la opción **Salidas** ubicada en la barra **Herramientas de componentes**.

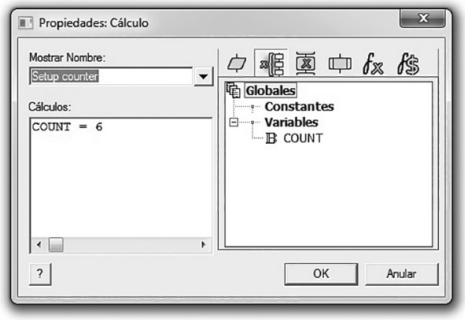


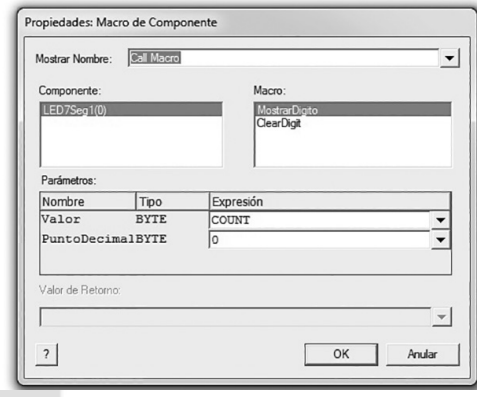
Para comenzar el montaje de manera física y a la par con el montaje lógico, es necesario tener a la mano la documentación correspondiente al fabricante y tipo de microcontrolador.

3. Posteriormente, se deberá hacer uso de la simbología, expuesta en el panel de herramientas de comandos correspondiente. Desde allí seleccionar los símbolos que se muestran a continuación y tratar de configurarlos.



4. Al dar doble clic sobre el símbolo insertado, se abrirá una nueva pantalla, la cual muestra los parámetros de configuración. Si se desea mostrar un número sobre el *display* antes invocado, deberá colocarse el valor declarado como variable, en este caso un número 6. A continuación, se muestra la ventana por cada símbolo insertado (*Setup counter* y *Call Macro*).





5. Para finalizar la prueba, basta con dar clic sobre el ícono **Play** ubicado en la barra de herramientas tradicional u oprimir la tecla **F5**. Si todo ha salido bien, el lenguaje se encarga de mostrar un número 6 sobre el *display*. Si se requiere cambiar el valor, esto se hace mediante las propiedades de cálculo (para lo que tiene que dar doble clic sobre el símbolo deseado).

Con este ejercicio será más sencillo comprender la creación de un algoritmo para la programación de microcontroladores, más adelante se describe el procedimiento completo para su puesta en marcha con ejercicios más elaborados.

8.4 PROGRAMACIÓN DE UN MICROCONTROLADOR

Antes de ingresar al tema de la programación de un microcontrolador PIC, es necesario tener en claro los elementos que se necesitan para tal tarea, estos son:

a. Un microcontrolador: Para la elección de este elemento, se debe tener en cuenta un conjunto de consideraciones como el propósito del proyecto, los puertos necesarios para ejercer un control ideal, en algunos casos el costo (aunque relativamente son económicos), el fabricante (pues este se encuentra en función de la versatilidad, pero también de la preferencia del usuario).

b. Un entorno de desarrollo: Donde se comenzará a ensamblar el código (instrucciones, diagramas de flujo, diagrama de procesos, etc.).

c. Derivado de lo anterior, debe considerarse el lenguaje de programación empleado para el IDE.

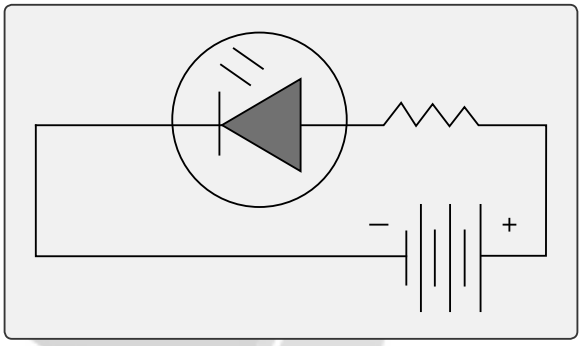
d. Y, por último, un kit de grabado que a menudo incluye el software necesario para efectuar esta tarea.

Para entender el contexto general de programación de un microcontrolador, se recomienda atender las siguientes consideraciones (expuestas en un paso a paso), acompañadas de un ejemplo real:

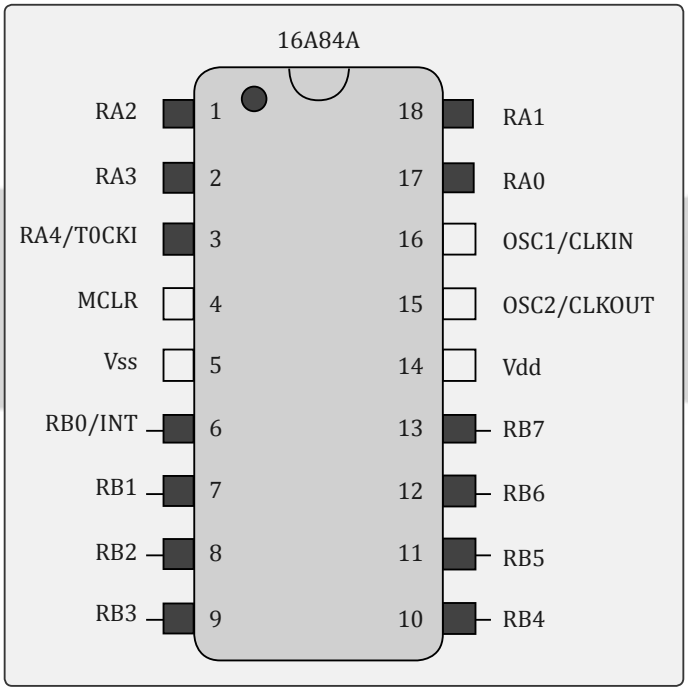
Uso de un microcontrolador para el control de luz de un led por un lapso de tiempo (led intermitente)

1. Lo primero que debe hacer es analizar a fondo el dispositivo (proyecto, prototipo, ejercicio) que será controlado por un microcontrolador.

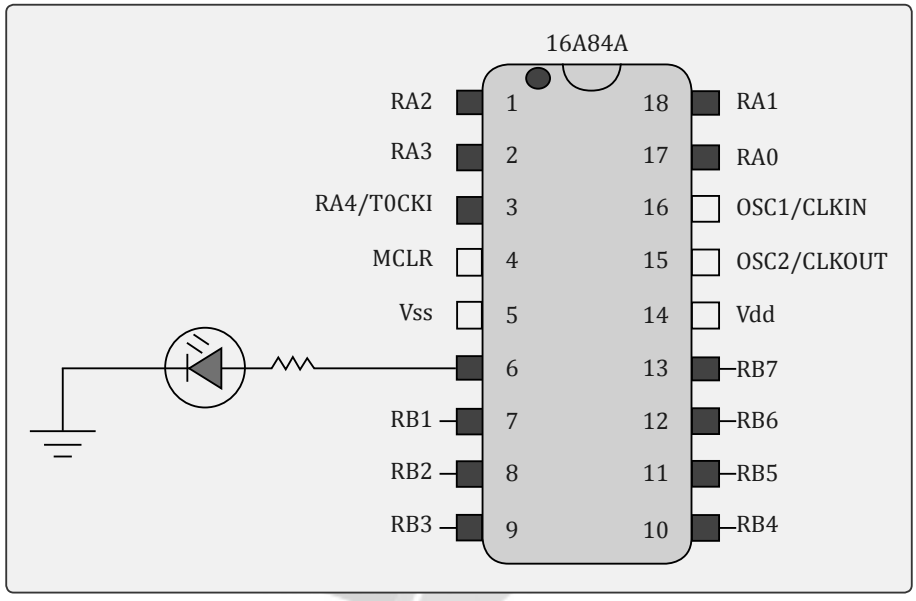
Para este caso se desea controlar el encendido y apagado de un led, dispuesto sobre una tablilla de conexión.



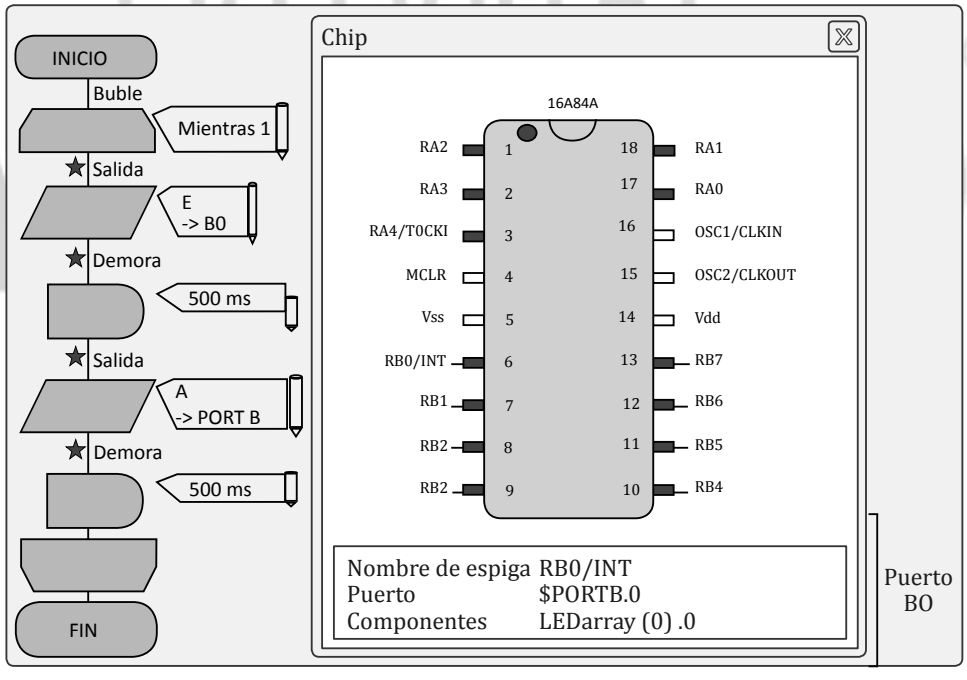
2. Haga un estudio sobre los microcontroladores disponibles y sus características (número de componentes). Esta elección se mantiene sujeta a las necesidades particulares del usuario. De una gran variedad de PIC, se tiene que elegir solo uno. Para este ejercicio, se empleará un microcontrolador 16F84A.



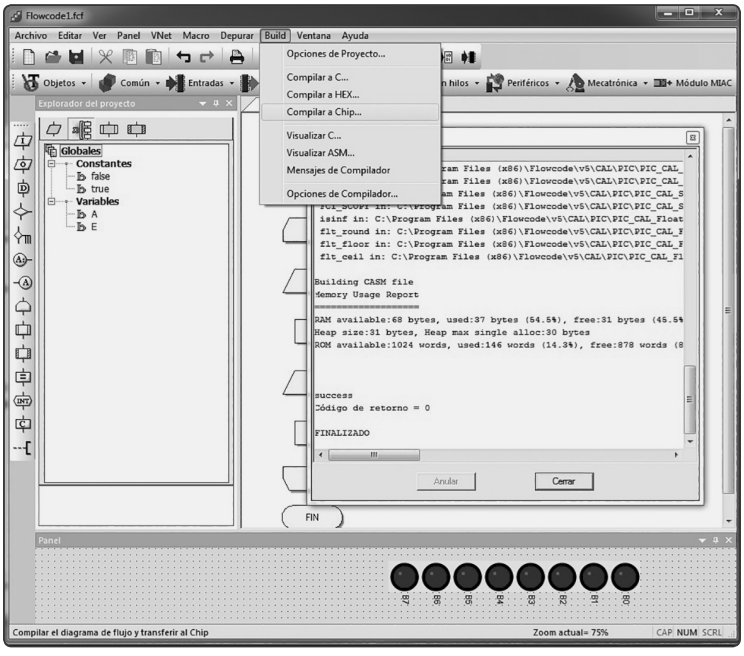
3. Diseñe (trace) y construya un prototipo que será controlado por el microcontrolador. Es necesario incluir los dispositivos periféricos que se utilizarán.



4. Debe tener a la mano una PC, además del lenguaje de programación compatible con el microcontrolador elegido. Se recomienda concebir el uso de herramientas de simulación para diseñar el sistema (representa los planos digitalizados del prototipo). En este caso, se empleará FlowCode como herramienta para programar el microcontrolador elegido mediante diagramas de flujo. Adicionalmente, se puede emplear algún simulador como Proteus.



5. Consiga un programador o quemador para grabar el código generado a la memoria del microcontrolador. Para efectuar la grabación, basta dar algunos clics dentro de la interfaz; para configurar el chip, compile el programa y después guárdelo en una ubicación segura dentro del disco duro.



El grabador de chips o microcontroladores se puede adquirir en alguna tienda de electrónica. La presentación de este producto es un kit, que por lo regular integra un cable USB, un ADP-023 (adaptador universal con cables Jumper), un par de conectores macho (SIL o *headers*) y, adicionalmente, un módulo ICSP que se puede adquirir de manera independiente o con interfaz USB integrada. Se compone de un zócalo compatible para el microcontrolador.

Para estas prácticas, se recomienda el uso de un programador PICKit 2 o 3 de Microchip para los chips de esta compañía). Es importante tener presente que todo grabador debe incluir su propia aplicación de software para efectuar la preparación, importación de archivos generados y grabación.

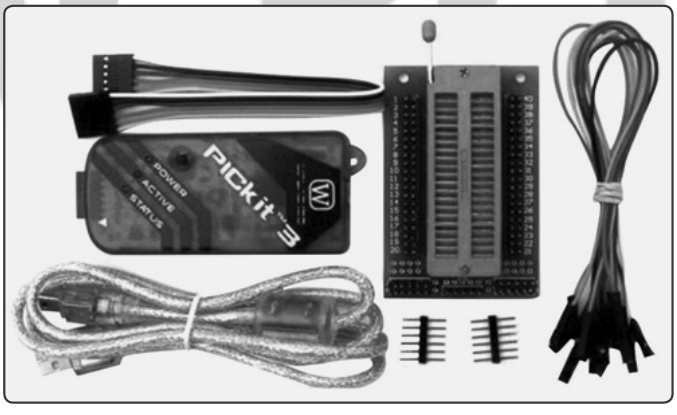
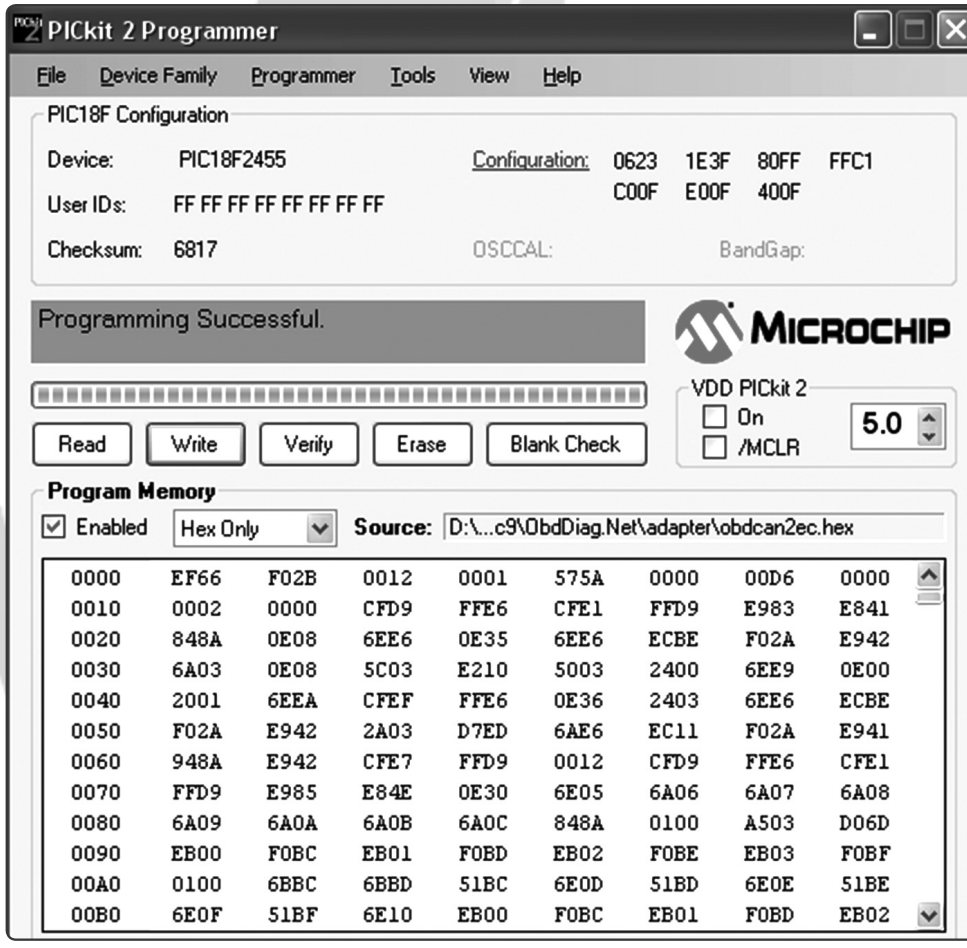


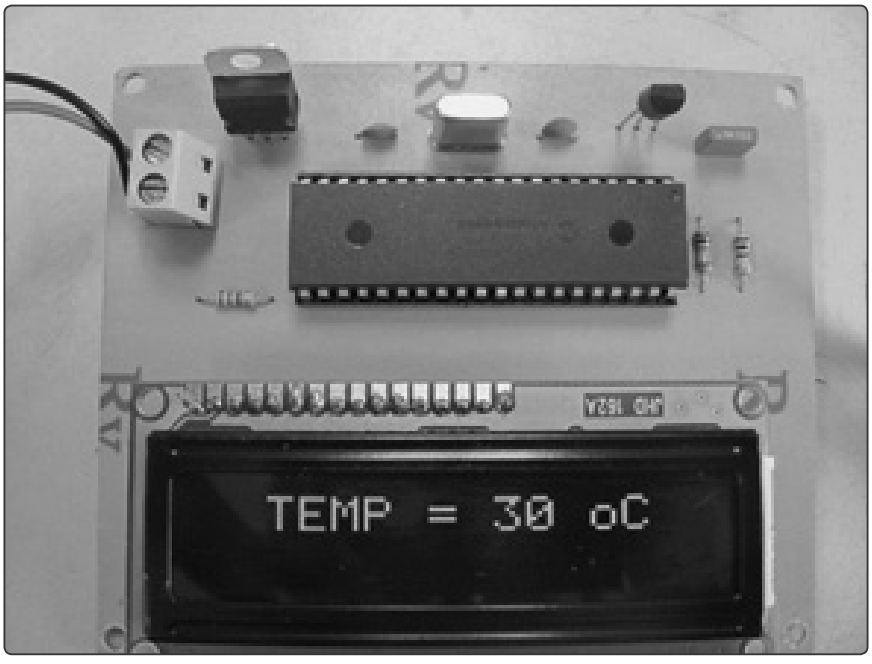
Figura 8.6 Kit grabador de microcontroladores

6. Una vez conectado el micro empleado (sobre protoboard o sobre el módulo ICSP) a la PC, se procede a grabar las instrucciones de FlowCode hacia el PIC. Para ello, utilice el software propio del kit adquirido (PICKit programmer). Desde allí se importa el archivo antes almacenado y se procede a la escritura del mismo al microcontrolador.

Nota: Para preparar la carga del programa generado desde FlowCode (o alguna otra herramienta) hacia el microcontrolador, se recomienda tener a la mano toda la documentación (*datasheet*, manuales) concerniente tanto del microcontrolador utilizado como del programador. Lo anterior con el fin de tener una referencia de conexión más certera. Desde luego, es importante salvar el código generado para poder importarlo (HEX) posteriormente.



7. Pruebe el funcionamiento del microcontrolador. Con el uso del módulo ICSP es posible efectuar pruebas de funcionamiento. No obstante, esta prueba, también se logra, montando el PIC sobre un protoboard, placa soldable o PCB.



Actualmente, existe una alta gama de microcontroladores y, por consiguiente, de fabricantes. Se recomienda visitar la página de los fabricantes donde se encuentra gran variedad de recursos, entre ellos están hojas de datos, esquemas de conexión, ejemplos, mejoras y recomendaciones.

ACTIVIDAD 1

1. Realice un programa que muestre una secuencia de ledes de color rojo. Asigne un tiempo de 500 milisegundos entre cada led.
2. Realice la misma secuencia de ledes, pero en orden ascendente y descendente.
3. Realice el montaje sobre el simulador Proteus para verificar su funcionamiento.
4. Realice la conexión física del punto 1 y 2 utilizando algún kit de grabación.
5. Investigue cómo grabar un programa desde FlowCode hacia el microcontrolador.

Práctica de laboratorio n.º 10

Microcontroladores PIC
Contador digital

Objetivo general: Introducir al usuario a la configuración y programación de un microcontrolador PIC.

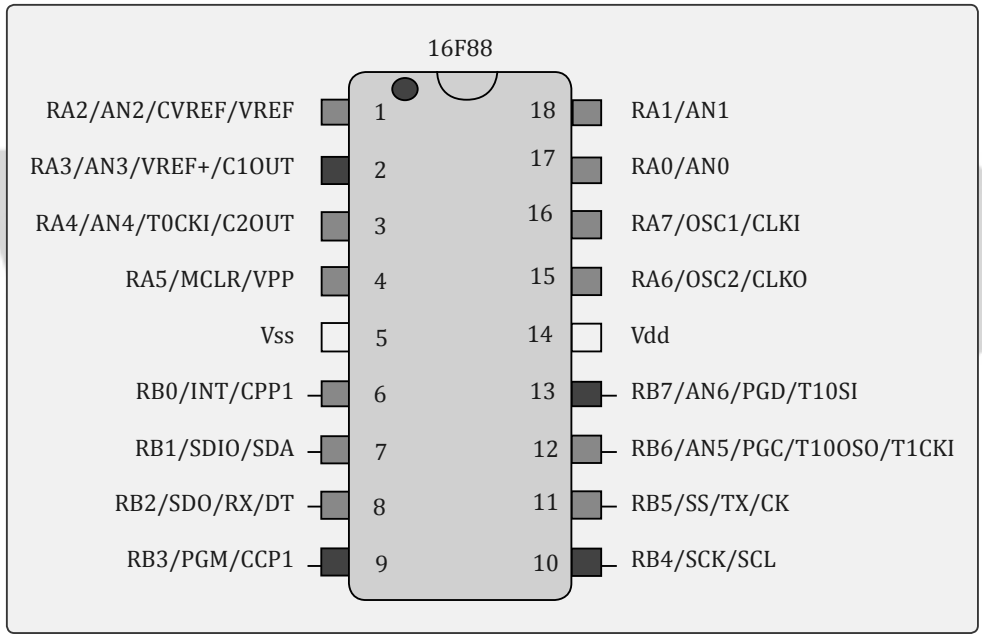
Objetivo específico: Aprender a configurar y programar un microcontrolador utilizando FlowCode para el diseño de un diagrama de flujo, capaz de mostrar un contador del 0 al 9 desde un display de siete segmentos.

Materiales:

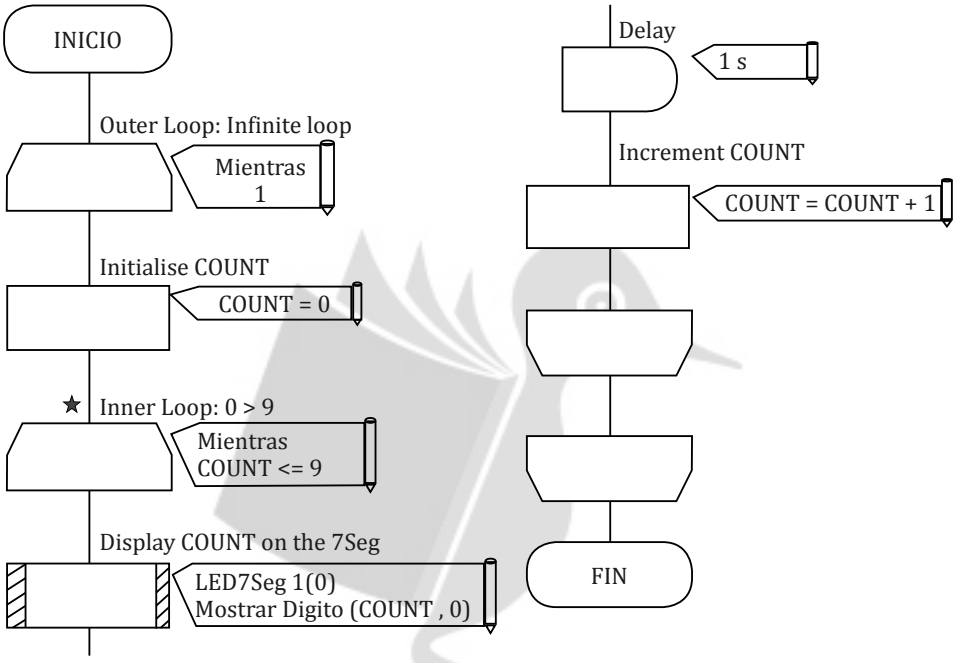
- Un microcontrolador 16F88.
- Un *display* de siete segmentos.
- El software FlowCode.

Procedimiento:

En la siguiente imagen, se muestra el esquema de distribución de pines del microcontrolador 16F88.



En la siguiente imagen se muestra el montaje del diagrama de flujo correspondiente al contador del 0 al 9.



a. Monte el diagrama de flujo sobre FlowCode, compílelo y ejecútelo para comprobar su funcionamiento.

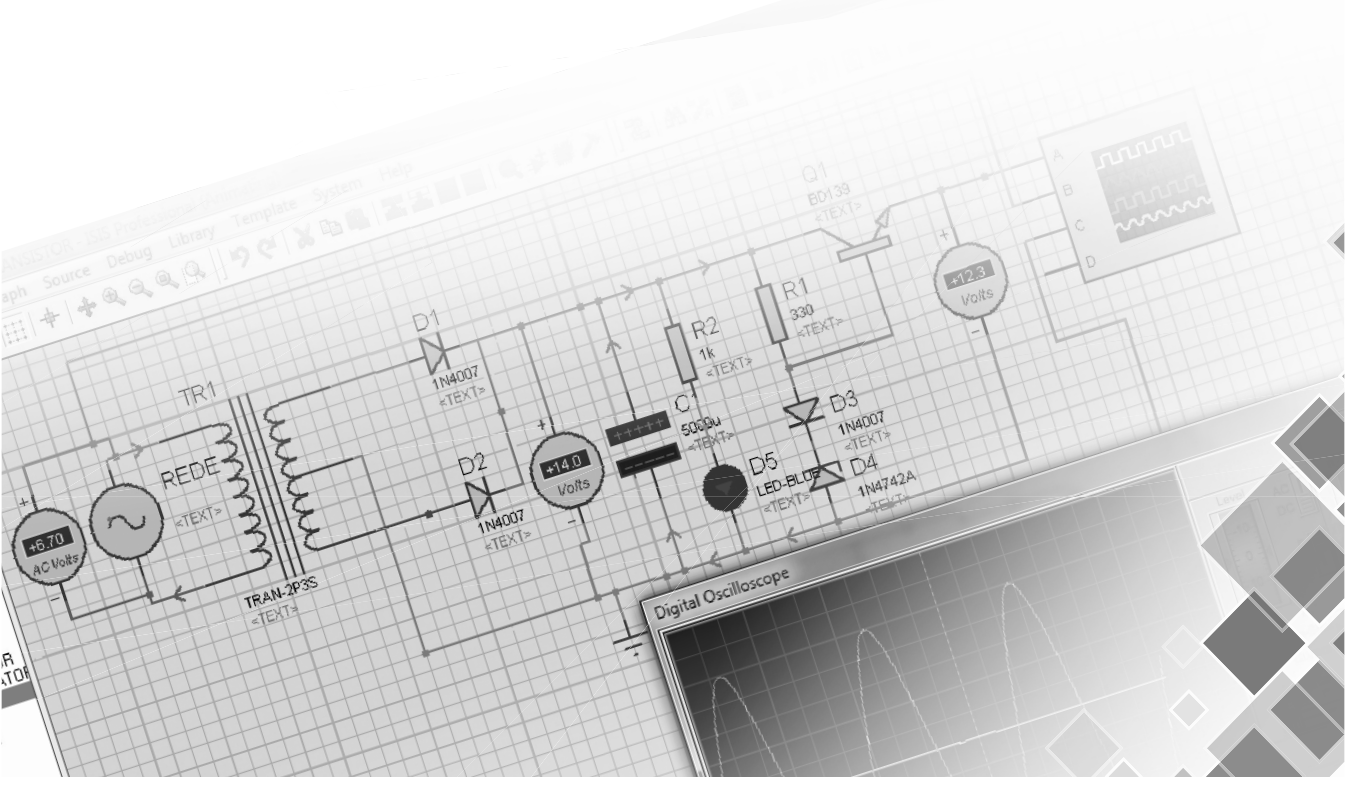
b. Arme un circuito físico sobre protoboard que permita la manipulación de este contador digital.

c. Cárguelo a un programador y pruebe su funcionamiento físico.

En este capítulo, se ha analizado aspectos importantes sobre los microcontroladores, revisando, inicialmente, los campos de aplicación, los elementos de software y hardware necesarios para su configuración, hasta la forma de programarlos y ponerlos a punto para su funcionameniento. Se ha descrito también, algunos ejemplos y prácticas sobre programación de estos pecualiares circuitos integrados através de FlowCode. En el siguiente apartado se presenta los aspectos generales sobre la creación de PCB.

Apéndice

A. Herramientas de software
B. Placas de circuito impreso (PCB)



EDITORIAL
MACRO®

APÉNDICE A: HERRAMIENTAS DE SOFTWARE

En este libro, mucho se ha hablado sobre herramientas de software que van desde utilerías de simulación hasta aplicaciones para la gestión, desarrollo e implementación de hardware. Inicialmente, este apéndice tiene la finalidad de orientar al usuario con respecto al manejo tanto de herramientas de simulación como para el desarrollo real de aplicaciones, módulos, prototipos, proyectos electrónicos o afines. Dejando al final las herramientas auxiliares para la documentación de ejercicios, prácticas o proyectos.

» HERRAMIENTAS DE SIMULACIÓN

Una herramienta de simulación permite poner a prueba un bosquejo o esquema de conexión antes de ser implementado. Lo anterior facilita, desde luego, la planeación del prototipo físico, reduce el tiempo en el proceso de montaje, permite la prueba de componentes, tensiones y herramientas, ofrece al usuario la oportunidad de depurar errores, además, de garantizar una conexión segura y sin accidentes. Actualmente, se desarrollan simuladores para el diseño lógico y montaje de esquemas de conexión, para la construcción de circuitos digitales y para la simulación de microcontroladores (como es el caso de Arduino).

Una herramienta de simulación representa la mayoría de las veces un recurso ideal para estudiantes y docentes que desean poner en practicar el diseño y la construcción de proyectos de distinta índole.

▪ Simuladores para diseño lógico

Para el diseño e implementación de circuitos digitales, a menudo se recurre tanto a la planeación como al trazado de un esquema de conexión que permita entender su funcionamiento. Ese es el objetivo principal de un diseño lógico. Existen muchas soluciones de software que permiten llevar cabo esta tarea. Algunas utilerías recomendadas son *Logisim*, *Logicly* (simulador en tiempo real online), *Digital Logic Design*, Atanua (herramienta de software libre, disponible para sistemas Windows y sistemas GNU-Linux y OSX).

A. *Logisim*

Se trata de una herramienta educativa para el diseño y simulación de circuitos lógicos digitales. Su interfaz es tan simple que facilita el aprendizaje de conceptos básicos relacionados con los circuitos lógicos, compuertas lógicas, diagramas lógicos, mapas de Karnaugh, análisis combinacional, además de permitir la construcción de circuitos complejos y de, prácticamente, cualquier sistema digital como registros, memorias, codificadores, multiplexores, etc. Habitualmente es usado para simular el funcionamiento de microprocesadores (CPU).

Logisim es una herramienta muy completa, pues integra el uso de *displays*, matrices, teclados y demás elementos que permiten la construcción de proyectos como juegos modulares y módulos de hardware. Es ideal para el montaje de compuertas lógicas.

B. *Logicly*

Es muy similar a la utilería anterior. Es ideal para el montaje en tiempo real, pues se encuentra disponible vía online. De igual modo cuenta con una interfaz sencilla para el manejo de compuertas y demás elementos. Permite arrastrar los componentes de manera dinámica e interactiva. Este recurso permite la ejecución del diseño paso a paso mediante el cual es posible verificar el flujo de las señales de un punto a otro. La página oficial de Logicly es http://logic.ly/.

C. *Digital Logic Design*

Se trata de una herramienta más para el diseño y simulación de circuitos digitales. Proporciona piezas digitales que van desde simples puertas a la unidad lógica aritmética. En este software, el circuito se puede convertir fácilmente en un módulo reutilizable. Un módulo puede ser utilizado para construir circuitos más complejos como CPU. El funcionamiento del circuito se puede analizar mediante el uso de piezas de salida como ledes, displays de siete segmentos, etc. Incluye componentes gráficos y una gran variedad de recursos. Esta utilería permite trabajar, incluso, con álgebra booleana.

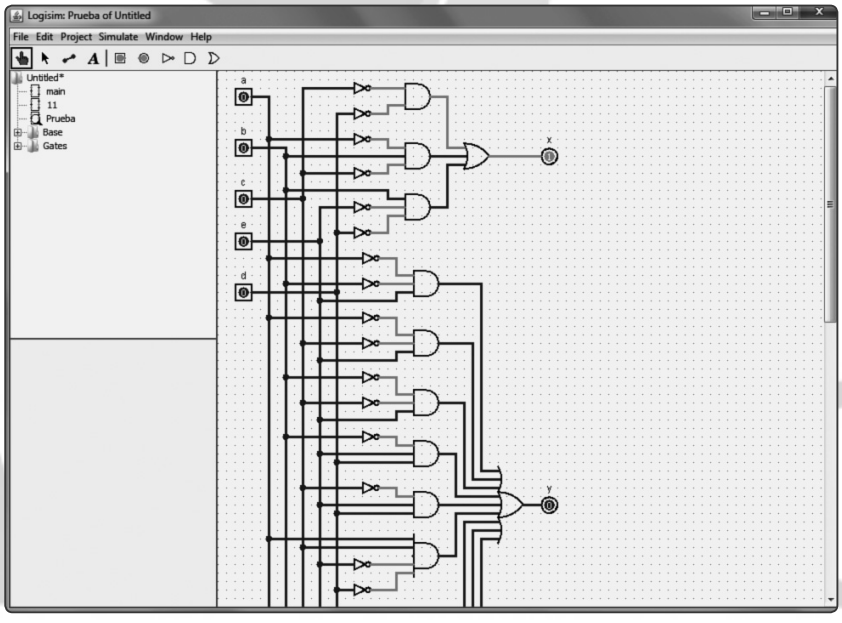


Figura A.1 Interfaz gráfica del simulador para diseño lógico, logisim
Fuente: el autor.

▪ Herramienta para simular circuitos electrónicos

Este tipo de recurso, a diferencia de las herramientas para diseño lógico que permite trazar circuitos lógicos, sirve para montar un diagrama topológico. Es decir, un circuito funcional sobre alguna placa de prueba (habitualmente un protoboard virtual) con el propósito de probar su funcionamiento y sus alcances. Es un previo a la implementación real. Las herramientas que se recomiendan son el constructor y simulador de circuitos digitales, *WinbreadBoard* y *LiveWire*/*PCB Wizard y Proteus.*

A. Constructor y simulador de circuitos digitales y *WinbreadBoard*

La interfaz de este simulador es muy sencilla e intuitiva. Posee una amplia área de trabajo donde se pueden arrastrar módulos integrados por componentes electrónicos y elementos de conexión como el protoboard, circuitos integrados, resistencias, pulsadores, interruptores (de dos estados), relojes, batería y *displays*. Este constructor gráfico incluye tutoriales, ejemplos para su montaje, escenarios y una gran variedad de chips.

Es ampliamente recomendable para estudiantes y profesores de las áreas de electrónica y la ingeniería de hardware.

En cuanto a *WinbreadBoard*, se sabe que su interfaz es muy parecida a la del constructor y simulador de circuitos digitales. Su entorno está integrado por un protoboard, un panel de *display* de siete segmentos, un panel de ledes, un *buzzer*, dos paneles de interruptores y una batería.

B. LiveWire / PCB Wizard

Es una aplicación más robusta que las antes descritas. *LiveWire*, por su parte es una interesante opción para trazar diagramas lógicos y a su vez diagramas topológicos. Cuenta con una gran variedad de componentes electrónicos bien clasificados para su uso. Una vez montado un circuito permite la simulación completa. Cuenta con interesantes ejemplos y montajes. Es una aplicación comercial de mucho auge en el mercado.

Por otro lado, se tiene a *PCB Wizard* que consiste en una aplicación complementaria de *LiveWire* para trazar el esquema de placas de circuito impreso (PCB). Es ideal para convertir un diseño lógico o topológico a un esquema PCB. El cual, al ser generado, puede imprimirse, para posteriormente implementarse.

C. *Proteus Design Suite*

Una de las herramientas más famosas para simular circuitos complejos, incluyendo microcontroladores, es sin duda *Proteus Design Suite*, o simplemente Proteus. Como su nombre lo indica se trata de una *suite* que incorpora una serie de recursos para efectuar tareas de índole electrónico (diseño de esquemas, montajes, programación de PIC, creación de PCB, etc.) Es la preferida por muchos usuarios y, aunque al principio parece compleja, es una herramienta versátil y sencilla de manipular. Es ampliamente recomendada para estudiantes y docentes. Aunque, para su manejo se requiere de conocimientos básicos de informática y electrónica.

▪ Simuladores para probar un microcontrolador

Proteus, aparte de ser una herramienta para simular el montaje de circuitos electrónicos, es un recurso para simular el trabajo con microcontroladores de manera real. Pero, también existen otras opciones muy interesantes, por ejemplo 123D *circuit simulator*, el cual permite el trabajo con Arduino de manera específica. Otra alternativa para el manejo de microcontroladores es el simulador *SimuProc* y *PIPPIN machine*, los cuales se describen más adelante.

A. 123D circuit simulator (Arduino)

El famoso simulador online de Arduino de Autodesk. Se trata de una herramienta muy poderosa que ofrece al usuario la posibilidad de montar un diagrama de conexión muy completo e interactivo sobre un laboratorio electrónico, pues posee gran variedad de componentes electrónicos y dispositivos. Sigue el mismo mecanismo para armar circuitos que los simuladores anteriores, pues basta con arrastrar el componente al área de trabajo, algunos ajustes y el diagrama estará listo en minutos. Ofrece también la posibilidad de trazar esquemas de conexión a la par del diseño de diagramas. Con esta herramienta se trazan PCB en tiempo real.

La potencia de *123D circuit simulator* es tal que hace posible la escritura de *sketch* dentro de la misma interfaz en una sección exclusiva para la inserción de código (llamada Editor de código). Desde luego, no puede faltar el monitor serial o ventana de monitoreo que permite la visualización de la información en pantalla.

Este simulador es una alternativa para quien no cuenta con la placa de Arduino. Los cambios a la interfaz pueden ser archivados y a menudo consultados vía online, e incluso pueden importarse proyectos. La página oficial es: https://123d.circuits.io/home/create.

Nota: Es importante recordar que, al tratarse de recursos online, el usuario deberá registrarse para obtener la posibilidad de interactuar con las herramientas consultadas. Desde luego que *123D circuit simulator* no es la excepción.

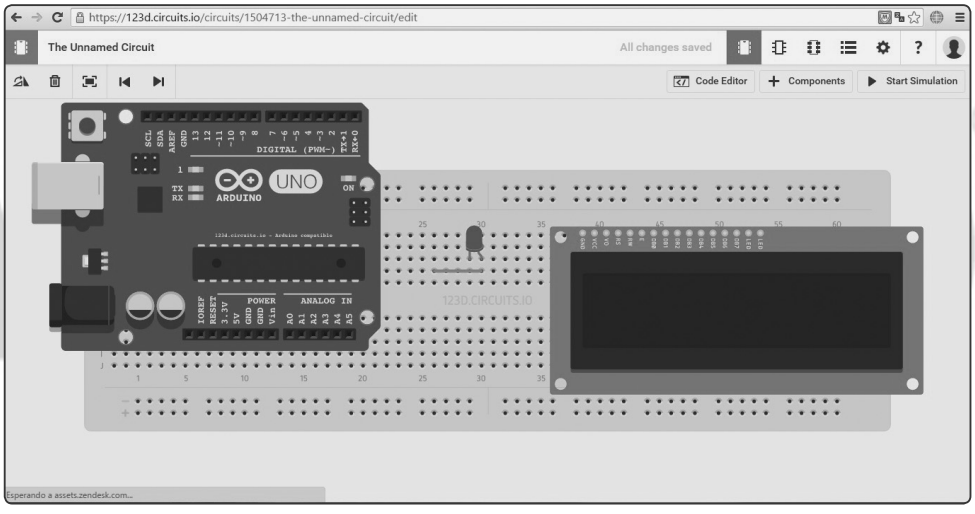


Figura A.2 Interfaz principal del simulador de Arduino 123D circuit simulator
Fuente: el autor.

B. *SimuProc y PIPPIN machine*

Ambos simuladores le permiten al usuario ser testigo del funcionamiento interno de un microprocesador. Son herramientas ideales en el aula. Su interfaz muestra la composición de una CPU y permite visualizar el proceso de almacenamiento, registro y procesamiento de la información. Ambos permiten interactuar con lenguaje ensamblador, código binario y hexadecimal. A menudo son utilizados como compiladores de lenguaje ensamblador. El *link* de descarga de *SimuProc* es http://simuproc.archivospc.com, y el enlace para obtener recursos sobre *PIPPIN machine* es http://www.dsie.upct.es/docencia/pippin.

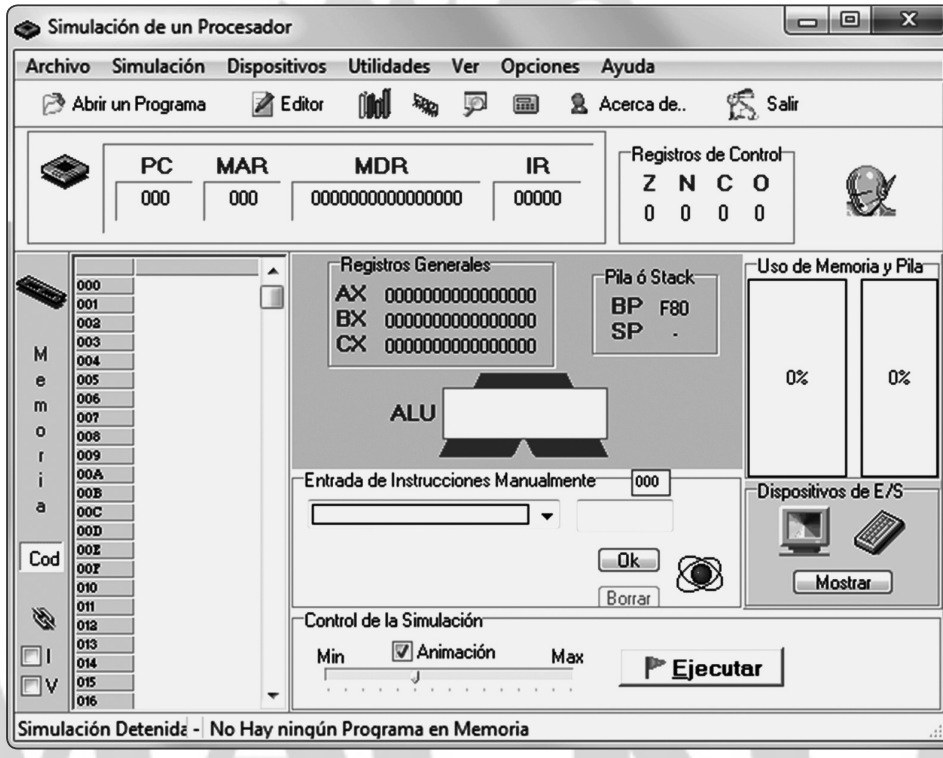


Figura A.3 SimuProc permite simular el funcionamiento de una CPU

» HERRAMIENTAS PARA LA DOCUMENTACIÓN DE PRÁCTICAS

Seguramente, en más de una ocasión, el usuario se ha visto en la necesidad de documentar alguna práctica, ejercicio o ejemplo para, posteriormente, publicarlo, imprimirlo o simplemente compartirlo. Hay varias soluciones de este tipo en internet, sin embargo, obedecen a un determinado sector o necesidad específica. Están integradas por herramientas para la construcción de diagramas o esquemas, gráficas, patrones de diseño, plantillas, tablas y demás. Una de las más famosas, cuando menos para la documentación de actividades de índole electrónica, es Fritzing.

▪ Fritzing

Esta popular aplicación contiene varias opciones para la personalización de diagramas de conexión. Fue desarrollada con la finalidad de publicar material en la web, fácil de entender, pues es muy llamativa e intuitiva. Representa una forma novedosa de invitar al usuario a la obtención de conocimientos. Es importante recalcar que *Fritzing* no es un simulador.

Esta eficaz herramienta permite el montaje a mano de circuitos de conexión que pueden incluir o no el uso de Arduino, microcontroladores, compuertas lógicas, sistemas digitales, fuentes, etc. Con ella se montan proyectos que desean mostrarse en el futuro a través de la web o de manera personalizada.

El sitio oficial de *Fritzing* es http://fritzing.org/home, donde se puede consultar información, proyectos desarrollados, kits de desarrollo, e incluso app sobre construcción de mapas para Arduino. Permite incluso la visualización de diagramas montados sobre protoboard, esquemas de conexión y PCB.

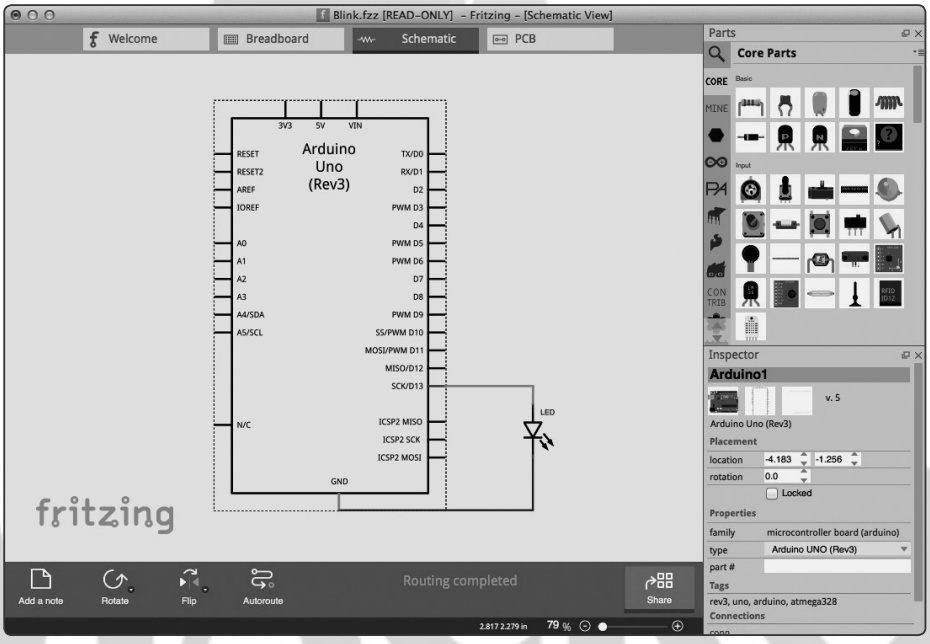


Figura A.4 Fritzing permite a veces la documentación de actividades desarrolladas en Arduino

» EL ENTORNO DE *PROCESSING*

Processing se define como un lenguaje de programación de código abierto, muy útil sobre todo para conseguir una comunicación entre el ordenador (o varias computadoras) y Arduino. Permite el almacenamiento de datos recolectados por Arduino. Bajo este lenguaje se encuentra desarrollado el IDE que la mayoría de los usuarios conoce. Permite la programación de entonos gráficos, además de la emulación de componentes de manera gráfica en tiempo real. Para obtener mayor información e instrucciones de su uso, se aconseja consultar el enlace http://playground.arduino.cc/Interfacing/Processing. Su portal oficial es https://www.processing.org.

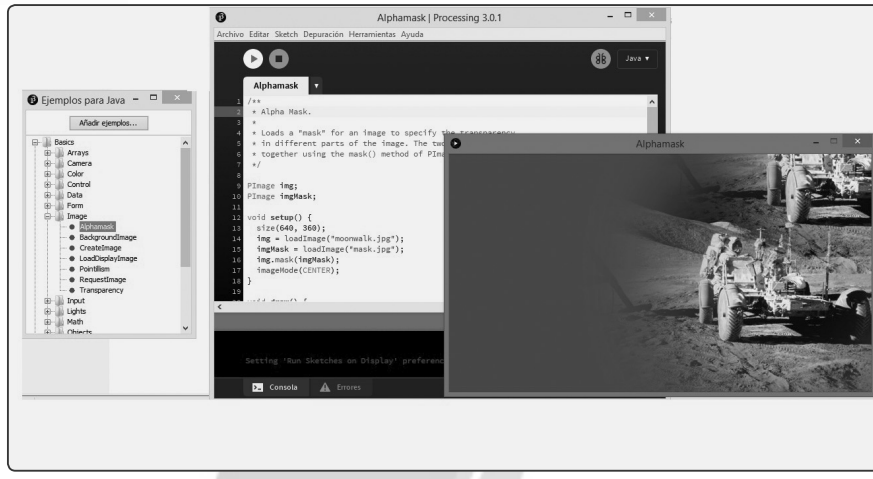


Figura A.5 Interfaz principal de Processing-Arduino

ACTIVIDAD 1

1. Mencione por lo menos tres herramientas de software para la construcción de diagramas lógicos.
2. Elabore un cuadro comparativo de, por lo menos, cinco herramientas de software para construir circuitos (diagramas topológicos).
3. Realice el montaje del siguiente ejercicio en *123D circuit simulator*.

Materiales:

- Un *display* de siete segmentos (cátodo común)
- Siete resistencias de 330 ohmios

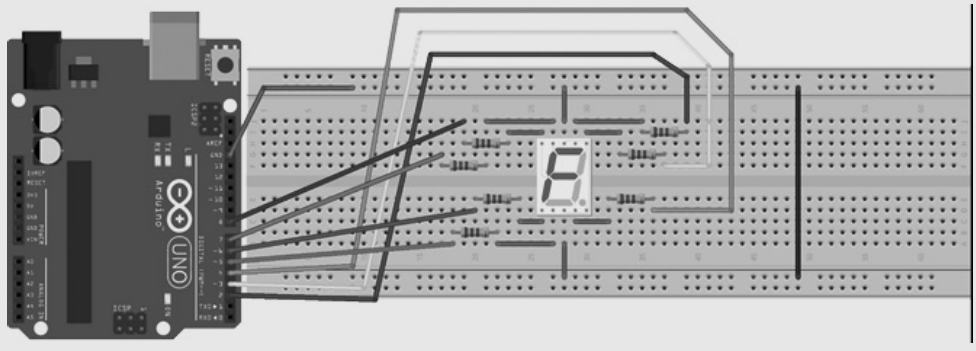
Realización:

El circuito permite mostrar nueve dígitos en un *display* de siete segmentos. Por lo tanto, se utilizará la función *void display* para establecer los parámetros que van a controlar el encendido de cada led o segmento. Se recomienda escribir cualquier dígito del rango antes señalado sobre el monitor serial después de cargar el código.

```
//Desarrollo de un sketch en Arduino capaz de mostrar 9 dígitos en un display de siete
segmentos
Int mensaje=0;
void setup(){

  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
```

```
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);

}
void display (int a, int b, int c, int d, int e, int f, int g)// Función del display
{
  digitalWrite (2,a);
  digitalWrite (3,b);
  digitalWrite (4,c);
  digitalWrite (5,d);
  digitalWrite (6,e);
  digitalWrite (7,f);
  digitalWrite (8,g);
}

void loop(){

  if (Serial.available()>0){
    mensaje=Serial.read();

    if (mensaje==’0’){
      display (1,1,1,1,1,1,0);
    }

    if (mensaje==’1’){
      display (0,1,1,0,0,0,0);
    }
    if (mensaje==’2’){
      display (1,1,0,1,1,0,1);
    }
    if (mensaje==’3’){
      display (1,1,1,1,0,0,1);
    }
    if (mensaje==’4’){
      display (0,1,1,0,0,1,1);
    }
    if (mensaje==’5’){
      display (1,0,1,1,0,1,1);
    }
    if (mensaje==’6’){
      display (1,0,1,1,1,1,1);
    }
    if (mensaje==’7’){
      display (1,1,1,0,0,0,0);
    }
    if (mensaje==’8’){
      display (1,1,1,1,1,1,1);
    }
    if (mensaje==’9’){
      display (1,1,1,0,0,1,1);
    }

  }
  delay(2000);
}
```

A continuación, se muestra el diagrama de conexión:



4. Realice la documentación en *Fritzing* del ejercicio desarrollado en el punto tres.

En esta sección se han analizado algunas herramientas de software muy útiles a la hora de trabajar con el diseño lógico, construcción de circuitos digitales y programación de microcontroladores. Por cada una se ha señalado su portal de internet, donde el usuario podrá consultar información al respecto, además de la opción a descargar interesantes recursos. En el siguiente apéndice se hablará sobre los métodos de construcción de PCB para la generación de placas para dar un acabado más profesional a ciertos proyectos.

APÉNDICE B: PLACAS DE CIRCUITO IMPRESO (PCB)

Como se ha mencionado, al diseñar un proyecto o prototipo electrónico, primero se debe probar armándose en una placa de prueba o protoboard. Cuando funcione correctamente, se debe trazar el esquema (ya sea a mano, o desde la PC) usando herramientas de software especializadas, las cuales se describirán más adelante en este apéndice. Posteriormente, se diseñará y fabricará el circuito impreso sobre el que se montan los componentes electrónicos. Finalmente, la placa generada puede colocarse sobre un gabinete que le dará una presentación final al proyecto realizado.

▪ Técnicas para la elaboración de PCB

Actualmente existen diferentes técnicas para efectuar el grabado de placas de circuito impreso (PCB) de manera manual o casera que, desde luego, tratan de imitar el proceso industrial. La diferencia a la hora de presentar un producto terminado está en los acabados, pues, evidentemente, la placa casera carece de profesionalismo. El uso de la técnica deseada dependerá de varios factores, como el presupuesto con el que cuenta el usuario y el propósito del proyecto/prototipo.

- Para la generación de pistas se utilizan técnicas como rotulado con tinta indeleble, adherencia de pistas, lápiz conductor.

- Para ejecutar el proceso de transferencia, las técnicas son por transferencia térmica, por insolación y revelado.
- Para protección y presentación de la placa, las técnicas son máscara antisoldante, estañado en frío, serigrafía.

A continuación, se describe el procedimiento general para el diseño de una PCB utilizando las técnicas antes citadas:

1. **Generación de pistas:** Se refiere a las técnicas empleadas para el trazado de las pistas que debe llevar el circuito. Deben trazarse de acuerdo al esquema lógico o topológico.
    - **Por rotulado con tinta indeleble:** Esta técnica es una de las más sencillas, aunque es ideal para el trabajo con pequeños esquemas que no requieren de un cableado complicado y extenso. La herramienta principal para el trazo de buses sobre la placa fenólica es un plumón con tinta indeleble, preferentemente de punto fino (aunque dependerá de los bordes del diseño). Este tipo de marcadores es resistente al agua y contiene acrílico.
    - **Por adherencia de pistas:** Actualmente se venden planillas de pistas y *pads* adheribles que tienen a bien formar un circuito para el ataque químico. Es una opción alterna al rotulado con tinta indeleble. Asegura un mejor acabado y mayor precisión. Se pueden encontrar en tiendas de electrónica.
    - **Por lápiz conductor:** Aunque poco se habla de este tipo de elementos, a menudo es utilizado para realizar trabajos de mantenimiento correctivo de hardware (reparación de pistas o estañado de contactos). Algunos productos novedosos ofrecen la posibilidad de trazar esquemas sobre las placas. Contienen generalmente tinta de plata.

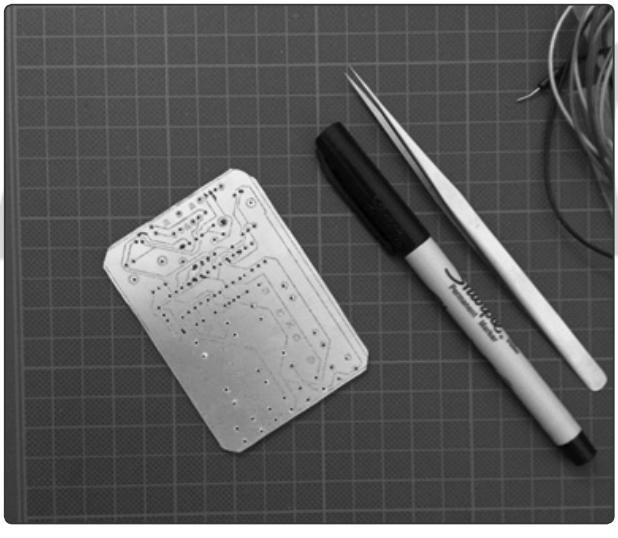


Figura B.1 El marcador de tinta indeleble, opción para remarcar un diseño o fotolito

2. **Proceso de transferencia:** Se refiere a la forma de ejecutar el grabado de pistas sobre la placa (independientemente de cómo fueron generadas). Esto se efectúa mediante calor (trasferencia térmica) o por exposición a la luz (insolación).

   - **Por transferencia térmica:** Esta técnica implica el manejo de herramientas de software especializadas (CAD) para la generación del diseño para su transferencia posterior hacia la placa fenólica. Una vez generado, debe ajustarse según sea necesario y, finalmente, debe ser impreso en una hoja de papel para su transferencia a través de calor (papel cuché, papel *transfer*). De este modo se tienen disponibles las pistas que formarán parte del circuito sobre placa. Es recomendable utilizar impresiones o fotocopias láser.

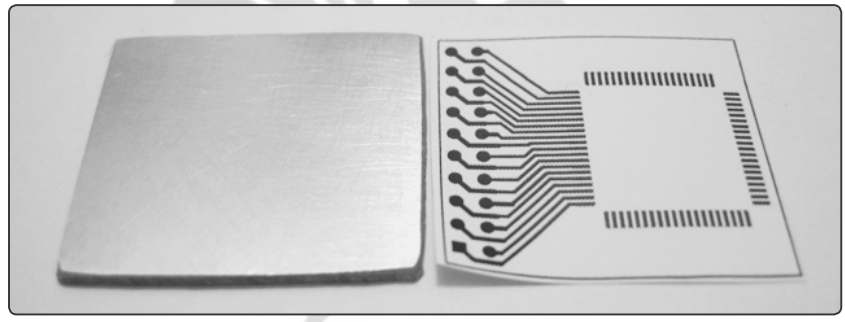


Figura B.2 Papel de transferencia para planchado

   - **Por insolación/revelado:** De igual modo implica el manejo de herramientas de software CAD para su transferencia a la placa mediante su exposición a la luz (por lo general luz UV, o luz negra). A menudo se utilizan hojas fotoresistivas, llamadas también transparencias fotosensibles, revelador líquido (soda cáustica o revelador universal sin hidróxido de sodio) y acetato para generar un fotolito[5] transparente. Incluso se puede comprar tarjetas fotosensibles para llevar a cabo esta tarea.

     La preparación de una PCB, tanto por transferencia como por insolación, requiere de un atacado químico, el cual, en términos de diseño y fabricación de hardware, se define como un proceso de corrosión del cobre sobre una superficie plástica o de polímero ante una emulsión. Lo anterior es posible gracias al uso de algún tipo de químico como sulfato de amonio, ácido clorhídrico, peróxido de hidrógeno o cloruro férrico, siendo este último uno de los más empleados en el desarrollo de PCB caseras.

[5] Un fotolito es un cliché (negativo) fotográfico impreso en papel transparente o translúcido como puede ser el acetato o el poliéster, en el cuál se encuentra la tipografía e imágenes que se desean serigrafiar. El fotolito se utiliza como plantilla para un posterior estampado fijándolo a la pantalla a través del proceso de insolación y revelado.

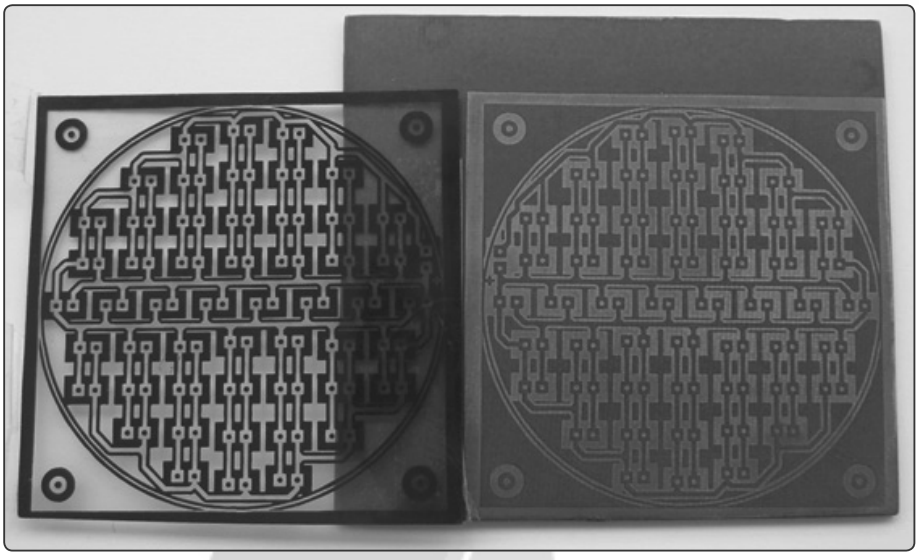


Figura B.3 La técnica de transferencia por insolación y el uso de placas fotosensibles y acetato

3. **Protección y presentación de la placa:** Consiste en el empleo de medidas para evitar la corrosión, exposición y ruptura de pistas, así como dotar de profesionalismo y presentación a la placa de circuito impreso.
    - **Máscara antisoldante:** Es una película protectora de pistas. Se encuentra disponible a modo de barniz y plantilla adherible. Protege toda la superficie de la placa, donde habitualmente se sueldan los componentes y reside el cobre.
    - **Estañado en frío:** Esta técnica consiste en el recubrimiento de pistas de cobre para su protección y estañado. Al momento en que la placa es expuesta a una emulsión, las pistas de cobre cambian su color a un equivalente en plateado. Para conocer otras técnicas profesiones consultar la página http://elektronikadonbosco.blogspot.com/2013/11/tecnicas-de-proteccion-y-acabado-final_15.html.

Nota: Se recomienda el uso del productor SUR-TIN, el cual consiste en un sistema de estañado por inmersión que protege la superficie de cobre de los circuitos impresos aumentando el agarre de las soldaduras. Con solo 2 minutos de aplicación, se deposita una capa uniforme y brillante de estaño sobre el cobre. Puede usarse varias veces y sirve para más de 2 m² de placa.

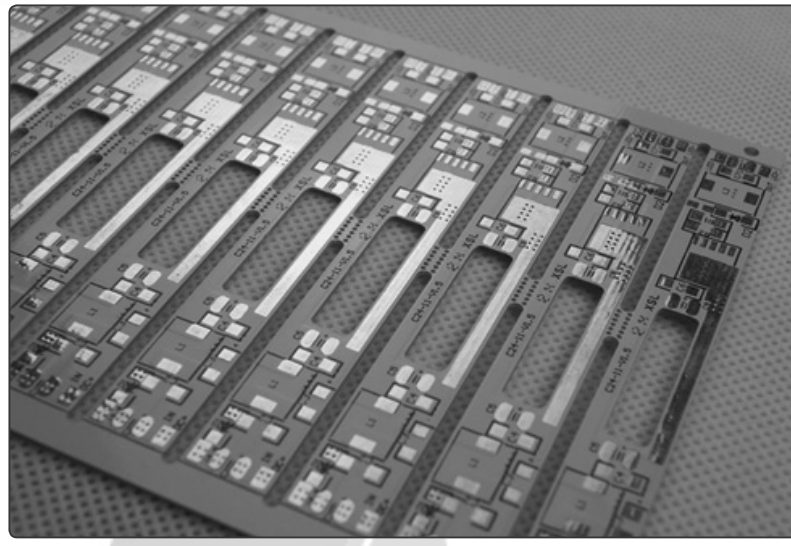


Figura B.4 Estañado de PCB, técnica de aislamiento y protección

- **Por serigrafía:** Para dar presentación a la placa. Esta técnica es con frecuencia utilizada para rotular la ubicación de los distintos componentes en la placa de circuito impreso, ya que su pintura no es conductora. Habitualmente se rotula de la parte donde se colocan los componentes. Requiere de un conjunto de materiales de serigrafía como son la pantalla (marco de madera con tela o malla), foto emulsión, insoladora (fuente de luz visible), paleta plástica, removedor de pantalla, limpiador universal, pinturas acrílicas para serigrafía, y estopa.

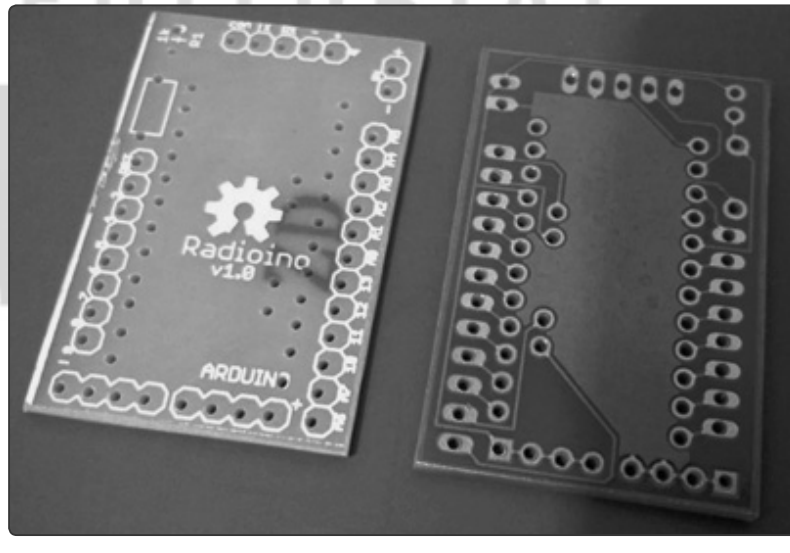


Figura. B.5 La máscara antisoldante garantiza un acabado más profesional a la PCB

▪ Herramientas de software para el diseño de PCB

Hoy en día existe una gran variedad de herramientas de software especializadas en auxiliar al usuario en el diseño de circuitos para PCB (y generación de fotolitos). Algunas de ellas son KiCAD, PCB Express, Proteus, *Eagle*, *Pspice*, *ZenitPCB*, *PCBWizard*, *PCBWeb Designer* y *Workbench*. A menudo, y recientemente, se utiliza *Fritzing*.

Se sabe que dentro del mundo de las PCB existe otra clase de consideraciones que hacen posible un verdadero trabajo de ingeniería y que garantizan mayor profesionalismo en el diseño. Se trata de métodos, fórmulas para efectuar cálculos de matemáticos, físicos y electrónicos, lo que trae consigo la consideración de herramientas de software complementarias. En la siguiente tabla se muestra el ejemplo de algunas de ellas:

Tabla B.1 *Herramientas complementarias en el diseño de PCB*

Nombre de la herramienta	Descripción
SwitcherCAD, *BodeCAD* y *FilterCAD*	Realiza esquemático, simulación, análisis en el tiempo y en la frecuencia. Diseño de Filtros Eléctricos/ Electrónicos
SciLab	Paquete científico para cálculos numéricos, realiza análisis en el tiempo y frecuencia (similar a MatLab)
SAPWIN	Esquemático y analizador de circuitos. Se obtiene expresión matemática del circuito propuesto
CircuitCalculator	Calcula el ancho de Pistas para PCB. Encuentra los valores comerciales de Resistencias y Condensadores

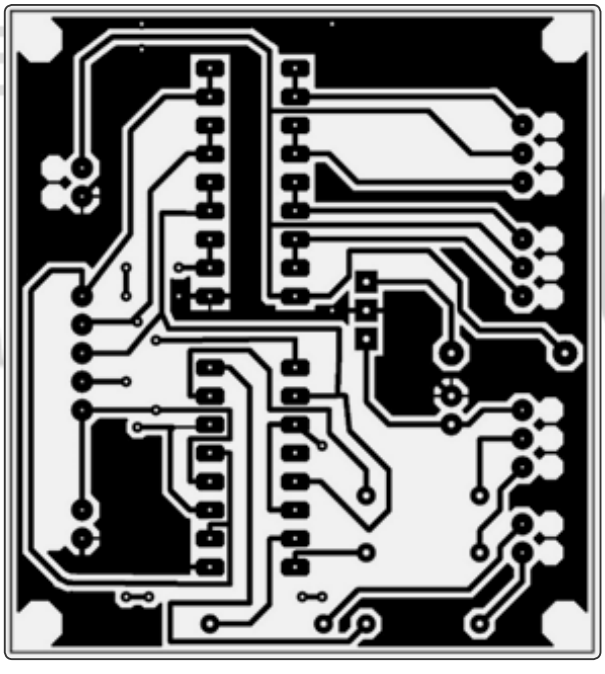


Figura B.6 La construcción de pistas y pads de una PCB se pueden generar gracias a herramientas de software CAD

» MATERIAL PARA LA CREACIÓN DE PCB

Uno de los materiales más utilizados para la fabricación de PCB es la placa fenólica, generalmente constituida de baquelita que consiste en un fenoplástico[6] resistente al calor y a los solventes, desarrollado por Leo Hendrik Baekeland. A menudo, se usa la fibra de vidrio con resina de poliéster en la fabricación de estas placas y, aunque es de mayor costo, suele ser de mejor calidad y presentación. Cualquiera de estos dos materiales están cubiertos por una delgada lámina de cobre en una o en ambas caras (dependiendo del tipo de placa, el cobre puede ir a su vez protegido por una capa de resina fotosensible). El cobre funciona como conductor eléctrico entre los distintos componentes que se montarán sobre la placa. Al momento de hacer un circuito impreso, el cobre de la placa tendrá trazado un conjunto de buses o caminos, los cuales se encargan de interconectar los componentes que irán en dicha tarjeta.

Los materiales empleados para la construcción de una PCB dependen de la técnica utilizada. Mientras que el material empleado para la construcción del circuito, depende totalmente del propósito del proyecto. A continuación, se citan los elementos necesarios para la construcción de una PCB. Con fines prácticos, se han separado en materiales para el diseño y herramientas auxiliares para su construcción.

a. Materiales para el diseño: En la siguiente tabla se muestra una lista de los materiales necesarios en caso de emplearse la técnica por transferencia térmica y transferencia por insolación, respectivamente.

Tabla B.2 *Materiales necesarios para el diseño de una PCB*

Técnica de transferencia	Transferencia térmica	Transferencia por insolación
PLACA	Placa fenólica virgen (baquelita)	Tarjeta fotosensible (o placa fenólica virgen + película fotorresistiva)
GENERACIÓN DE PISTAS	Marcador de tinta indeleble, pistas y *pads* adhesivos, lápiz conductor	
PAPEL	Papel *transfer* o cuché	Acetato para fotolito
MATERIAL / TRANSFERENCIA	Pistola de calor o plancha	Lámpara de luz UV
ATACADO QUÍMICO	Cloruro férrico	Cloruro férrico

[6] Masa plástica prensada hecha con resina artificial compuesta por fenol y formaldehído.

b. Herramientas adicionales para su construcción: Son herramientas auxiliares para concretar el acabado de la placa de circuito impreso. Algunas de estas son las siguientes:

- **Taladro o dremel (mini taladro) con accesorios:** Contiene puntas intercambiables, discos abrasivos y de metal, fresas y su respectivo cargador DC. Los kits para PCB son ideales para llevar a cabo esta tarea.
- **Brocas para dremel o taladro manual (alternativa):** Estas pueden variar su tamaño, aunque las más recomendables son de 0.7 y 0.8 mm.
- **Cautín de lápiz:** Permite soldar y desoldar componentes electrónicos. Se recomienda el uso de cautines con puntas intercambiables de 20 a 30 W con la posibilidad de aumento hasta 200 W. De preferencia cautines reguladores de temperatura con soporte para el lápiz.
- **Pistola de calor:** Ideal para transferencia térmica o plancha.
- **Lámpara de rayos UV:** Se puede fabricar o comprar en alguna eléctrica.

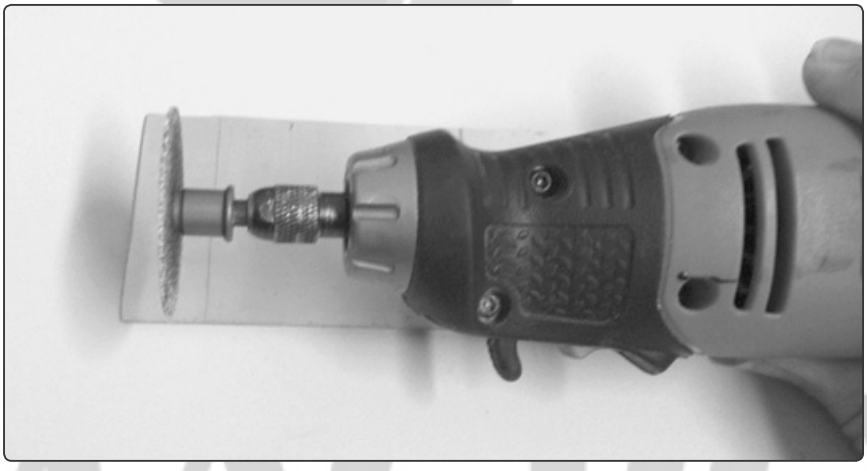


Figura B.7 El dremel, herramienta fundamental para efectuar cortes, pulido y perforación de PCB

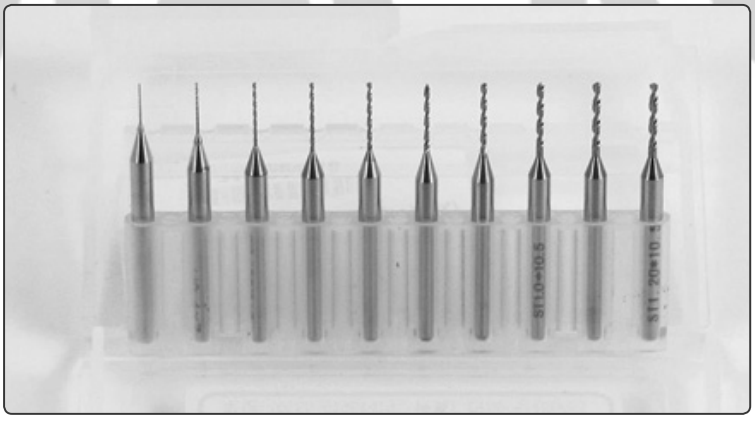


Figura B.8 Las brocas especiales para PC recomendadas son de 0.7 y 08 mm

» CONSIDERACIONES Y RECOMENDACIONES GENERALES PARA EL DISEÑO DE PCB

- Efectuar un trabajo como el diseño de una placa de circuito impreso, requiere a menudo de un entorno o lugar adecuado con suficiente ventilación o con extractores de aire.
- Se recomienda usar tapaboca, guantes y lentes protectores. Evitar tallarse los ojos, sobre todo cuando se tiene contacto con algún químico.
- Es recomendable tener conocimiento previo en el manejo de los químicos que se van a utilizar. De ser necesario, habrá que leer las indicaciones de uso, impresas en el propio envase.
- Verificar que la impresión en el papel esté correctamente orientada. Esto evita tener que realizarlo en repetidas ocasiones.
- Realizar el corte de la baquelita antes de comenzar el grabado del diseño impreso (desde la herramienta de software se determina el tamaño que tendrá el diseño).
- Verificar que la superficie de cobre no presente imperfecciones (suciedad, cortes, rayaduras). Por eso es necesario lijar, desengrasar y limpiar con solventes (acetona), antes de grabar el diseño.
- Se recomienda el uso del marcador de tinta indeleble para corregir los problemas con las pistas o *pads* del impreso. Lo anterior se debe realizar antes de sumergir la tarjeta en el ácido empleado.
- Tomar en cuenta el tiempo de oxidación del cobre en función del reactivo utilizado. Esto puede llevar minutos. Por lo tanto, se recomienda mover constantemente el refractario donde se aloja la placa.
- Se debe saber que la separación mínima entre 2 pistas adyacentes debe de ser 0.8 mm, lo que garantiza un buen aislamiento eléctrico de hasta 180 voltios, en condiciones normales.
- Realizar los *Pads* (Área plana conductiva donde se realiza las soldaduras de componentes) en función del tamaño y peso de los componentes. Así como también de acuerdo a las fuerzas y tensiones mecánicas que deba soportar.
- Es recomendable que los componentes puedan cambiarse sin necesidad de extraer otro.
- Se recomienda utilizar brocas de 0.7 a 1.25 mm de diámetro (se pueden adquirir kits de brocas).

Nota: La medida ideal para casi cualquier componente electrónico es de 0.7 y 0.8 mm de diámetro. Se pueden solicitar incluso como brocas de 1/32.

- Colocar y ubicar los componentes en la dirección y posición correcta. Esto evitará que el componente no coincida con la perforación.

- Escoger el tipo de estaño adecuado. Existen por lo general aleaciones 60/40 (estaño y plomo, respectivamente) que solo necesitan 190 °C para fundirse.

Nota: El hierro de las terminales de una resistencia funde a más de 1500 °C. El cobre de una baquelita a 1080° C. El estaño puro funde a 232 °C y el plomo a 327 °C.

Ya que funde a 300 grados. Tratar de utilizar estaño con alma de resina o colofonía para soldar (lo que evita el uso de pastas para soldar).

- Se recomienda tener conocimiento previo sobre soldadura. Lo que garantiza un trabajo con mayor calidad y menor tiempo de diseño. Nunca soplar hacia la soldadura.
- Se recomienda colocar el soldador en un ángulo de 45 grados. Se recomienda iniciar soldando los componentes más pequeños y, de ser posible, de adentro hacia afuera.
- Luego de soldar todos los componentes, es recomendable utilizar sustancias protectoras o máscaras antisoldantes.

» PROCEDIMIENTO GENERAL PARA EL DISEÑO DE PCB

A continuación, se describe el procedimiento general para el diseño de placas de circuito impreso. Desde luego, se hace hincapié en ambas técnicas de transferencia antes mencionadas.

- **Diseño:** Un circuito complejo requiere el uso de herramientas computacionales que permitan diseñar y simular el esquema electrónico y su arte de circuito impreso (para trabajos sencillos se puede efectuar el trazo con marcador indeleble o planillas adheribles).

Nota: Es importante configurar el software utilizado para arrojar la impresión del diseño PCB en modo espejo.

- **Corte:** Una vez que se conoce el tamaño del circuito impreso, se procede a realizar el corte de la tarjeta. A menudo se utilizan seguetas y discos de corte fino (abrasivos o de metal para dremel).
- **Impresión sobre papel:** Una vez que el diseño está listo, se planea la impresión sobre el papel deseado. Esto depende del tipo de transferencia que desea emplearse.
- **Transferencia en la placa de cobre:** Se procede a grabar el esquema o fotolito sobre la placa, este procedimiento depende de la técnica a utilizar (transferencia térmica o insolación).

TRANSFERENCIA TÉRMICA	TRANSFERENCIA POR INSOLACIÓN
Pasos para la transferencia de la imagen del papel *transfer* a la placa de cobre	Pasos para la transferencia de la imagen del acetato a la placa fotosensible
• Imprimir la imagen en el papel *transfer*. Usar láser o fotocopia • Cortar la baquelita al tamaño de la imagen y lijar las imperfecciones (usar fresa de dremel) • Limpiar, desengrasar y secar la baquelita • Colocar la placa con la cara de cobre hacia arriba; después, colocar la imagen del papel *transfer* con el lado de impresión láser en contacto directo con la superficie de cobre • Aplicar calor, ya sea con el uso de una pistola de calor o plancha. Verificar, continuamente, que el tóner se adhiera lo mejor posible a la placa • Sumergir el papel con la baquelita en una bandeja de agua tibia y realizar presión sobre el papel en la placa. Retirar lentamente el papel	• Imprimir con impresora láser o fotocopiadora la imagen sobre el acetato (se origina un fotolito) • Cubrir la placa con la película o transparencia fotosensible. Colocar el lado de impresión del tóner en contacto directo con la resina protectora de la superficie de cobre • Colocar la placa en la insoladora (emisora de luz UV). El tiempo de exposición a los rayos UV depende de la sensibilidad de la placa. Pueden ser segundos o minutos • Retirar la transparencia fotosensible y sumergir la placa en el revelador líquido. • Lavar con abundante agua

- **Atacado del cobre:** Se inserta la placa de cobre, previamente grabada, en soluciones ácidas que eliminan el cobre no deseado.
- **Limpieza y perforación:** Se realiza el lavado y limpieza de la placa para eliminar todas las impurezas, luego se perforan los orificios en donde se colocarán los componentes.
- **Soldadura:** Etapa donde se realiza el montaje (colocación y soldadura) de los componentes.
- **Pruebas de Funcionamiento:** Antes de realizar interconexiones se verifica el funcionamiento del circuito.

Figura B.9 El cloruro férrico, químico muy utilizado para la corrosión de cobre en PCB

» TECNOLOGÍA DE MONTAJE SUPERFICIAL (SMT)

Los chips de nueva generación poseen una tecnología llamada tecnología de montaje superficial (SMT), la cual les permite ser retirados con solo calentar sus pines que se encuentran haciendo contacto con la superficie de la placa. Lo que significa que la placa no está perforada como se hacía en la tecnología *throughhole*.

La tecnología SMT se incorpora en aparatos y equipos nuevos, sobre todo en computadoras, pues resulta una buena opción para reducir espacio. Son muy comunes también en placas de software y hardware libre, en teléfonos celulares, tabletas y todo tipo de laptops. E incluso, el servicio de mantenimiento correctivo a menudo exige tener que reparar o sustituir algún componente electrónico (dispositivo de montaje superficial o SMD) de alguna placa de circuito impreso, por lo que para ello es necesario tener que retirar la pieza para su análisis. Los aditamentos más comunes para proceder con esta tarea son:

- Cautín de lápiz preferentemente.
- Flux líquido (de preferencia con plata).
- Barritas para desoldar.
- Malla desoldadora u otra herramienta de succión de soldadura.
- Pinzas de precisión.

El método general para desmontar dispositivos de montaje superficial se ilustra en el siguiente esquema:

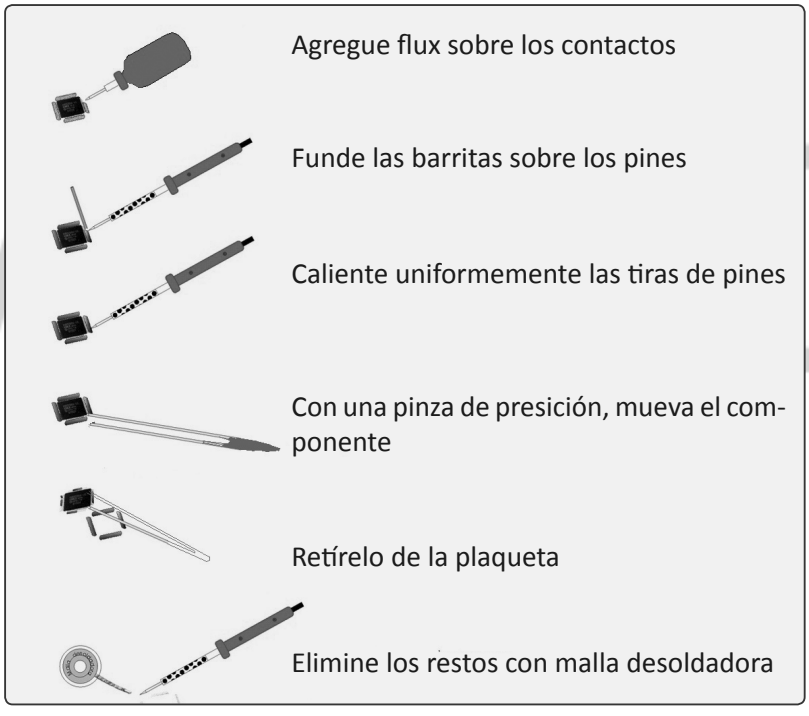


Figura B.10 Procedimiento básico para desmontar un componente de cualquier PCB. Para volverlo a montar, bastará con soldar sus contactos a la placa

- ***Reballing***

Ciertos centros especializados en mantenimiento correctivo de equipos, y en específico de computadoras, hacen uso de estaciones especiales llamadas estaciones de *reballing*. Esta herramienta consiste en una planta especializada para trabajos de soldadura que incluye la tecnología de montaje superficial (SMT). Esta estación, generalmente está integrada por una lámpara de rayos infrarrojos que se encarga de aplicar calor a los distintos circuitos y se pueda montar y desmontar.

Otro tipo de herramientas usada en mantenimiento correctivo es la tina ultrasónica. Se trata de un elemento que permite la limpieza profunda de algunos módulos de la PC como tarjetas periféricas, módulos de memoria RAM e inclusive la propia placa base. Cuando se adquiera una tina ultrasónica, es recomendable que esta sea lo suficientemente espaciosa como para alojar una *motherboard*. Es importante también considerar la compra de agua destilada y algún detergente líquido casero.

Las diferentes empresas del mundo, pero sobre todo las que se dedican a la fabricación de hardware, poseen máquinas industriales de alta potencia y precisión, capaces de poder desarrollar cientos de módulos o productos en minutos, e incluso, algunos son colocados a mano para posteriormente ser soldados. En el siguiente enlace se muestra el procedimiento que se sigue en Taiwán para la construcción de *motherboard* para computadoras, este es http://jhonusuriagamainboard.blogspot.mx.

Figura B.11 Estaciones reballing, usadas por los técnicos en reparación y sustitución de dispositivos de montaje superficial, aunque su precio es bastante elevado

» PRUEBA FINAL DE UN PROTOTIPO

Para ejercer esta prueba se requiere de una punta lógica (detector de niveles de tensión) o *tester* (multímetro). A menudo es utilizado para la medición. Es una herramienta que no puede faltar en ningún taller de electrónica, robótica o mecatrónica. Se recomienda el multímetro digital, pues es más preciso que su antecesor, el multímetro analógico. Ambos cumplen con la función de medir el voltaje, la corriente y la resistencia de cualquier módulo o componente electrónico. Es importante recordar que, prácticamente, todos los dispositivos que integran un equipo, proyecto o prototipo, se hallan constituidos internamente de elementos como chips, capacitores, resistencias, condensadores, transistores, etc. y que, la mayoría de veces, las fallas más representativas provienen de ese tipo de elementos. El multímetro permite conocer con mucha exactitud la tensión proveniente de cualquier componente electrónico.

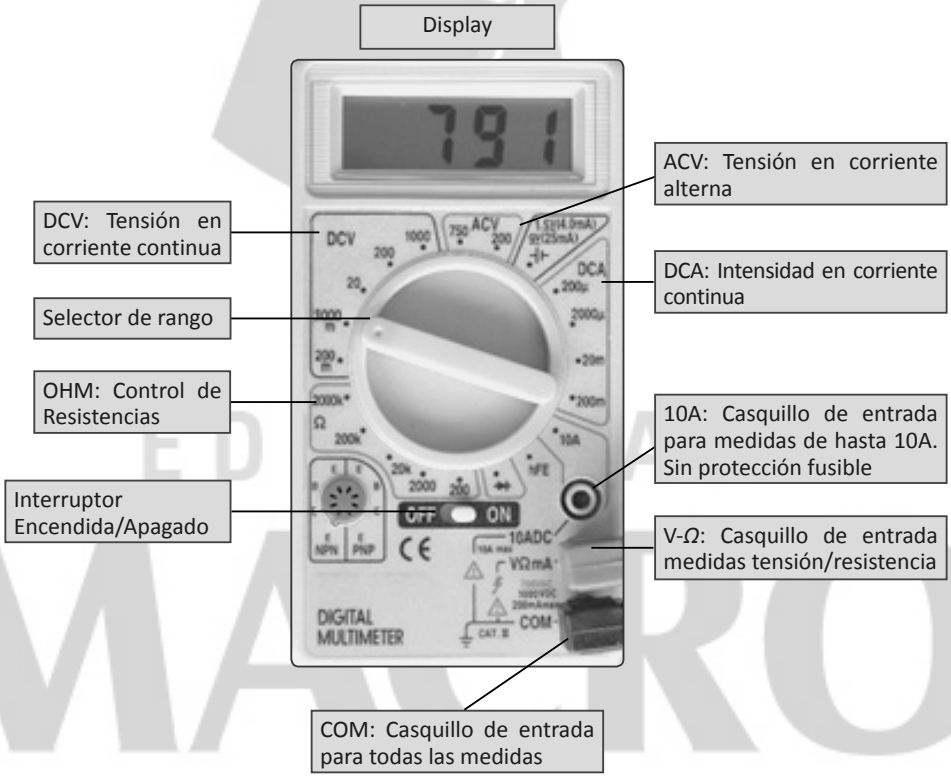


Figura B.12 Diagrama de las partes de un multímetro digital

En este último apéndice se han analizado aspectos importantes sobre las placas de circuito impreso (PCB) desde su planeación, diseño, construcción y mantenimiento. Se han dado a conocer los materiales necesarios para comenzar a trabajar y se han descrito los pasos que se deben seguir para convertir un simple prototipo montado en una placa de prueba a una PCB para su presentación, reproducción o venta.

Bibliografía

- Aranda, D. (2014). *Electrónica, plataformas Arduino* y *Raspberry Pi*. Buenos Aires, Argentina: RedUsers.
- Barden, W. (1986). *Matemáticas para programadores*. Madrid: Anaya Multimedia.
- Chávez, P. (2014). Compendio de lógica. México: Patria.
- Floyd, T. (2006). *Fundamentos de sistemas digitales*. 9.ª ed. Madrid: Prentice Hall.
- Floyd, T. (2008). *Dispositivos electrónicos*. 8.ª ed. México: Pearson.
- González, G. (2012). *Servicio técnico: Notebooks*. Buenos aires, Argentina: RedUsers.
- Quiroga, P. (2010). *Arquitectura de computadoras*. Madrid: Alfaomega.
- Romero, C., Vazquez, F. y De Castro Lozano, C (2010). *Domótica e inmótica*. *Viviendas y edificios inteligentes*. (3.ª ed.). Madrid: RA-MA.
- Rossano, V. (2009). *Electrónica y microcontroladores PIC*. Buenos aires Argentina: Editorial RedUsers.
- Ruiz, E. (2007). *Educatrónica: innovación en el aprendizaje de las ciencias y la tecnología*. Madrid: Díaz de Santos.
- Tojeiro, G. (2014). *Taller de Arduino*. Madrid: Alfaomega.

Impreso en los talleres gráficos de

EDITORIAL MACRO®

Surquillo