Apuntes de Microprocesadores y Microcontroladores

Descripción breve

Este es un resumen sobre Microprocesadores y Microcontroladores, con fundamentos de lógica digital y ejemplos de programación.

Revisión: marzo 2024, SPS. Honduras

Ricardo Adonis Caraccioli Abrego

ricardocaracciolihn@gmail.com

Versión: marzo del 2024

Prefacio

En este resumen se presenta lo necesario para poder guiar a un estudiante de la clase de microprocesadores y microcontroladores en los fundamentos necesarios para poder comprender el tema. Se abordan desde los conceptos básicos de la lógica digital hasta la programación de microcontroladores modernos y el uso de interfaces hombre–máquina.

Índice general

1 Lógica digital **1**
- 1.1 Lógica combinacional . . . 1
  - 1.1.1 Compuertas lógicas . . . 1
  - 1.1.2 Teoremas del Álgebra Booleana . . . 4
  - 1.1.3 Mapa de Karnaugh . . . 7
  - 1.1.4 Circuitos MSI . . . 9
  - 1.1.5 Circuitos Aritméticos . . . 12
- 1.2 Lógica secuencial . . . 15
  - 1.2.1 Flip-Flop . . . 15
  - 1.2.2 Detector de flancos . . . 15
  - 1.2.3 Contador binario . . . 16
  - 1.2.4 Registros . . . 17
  - 1.2.5 Memorias . . . 17

2 Microprocesadores **19**
- 2.1 Arquitecturas de computadoras . . . 19
- 2.2 ALU . . . 20
- 2.3 Buses . . . 20
- 2.4 Microprocesador . . . 21
- 2.5 Ejemplos de Microprocesadores . . . 21
- 2.6 Lenguaje de máquina, ensamblador y de alto nivel . . . 26
- 2.7 Tarjeta Madre . . . 27

3 Microcontroladores **28**
- 3.1 PIC . . . 28
- 3.2 ATMEGA . . . 30
- 3.3 Comparación entre Microcontrolador y Microprocesador . . . 31

3.4 Tarjeta Microcontrolador . . . 31
3.4.1 Entradas y salidas discretas con Tarjeta Microcontrolador . . . 33
3.4.2 Entrada analógica con Tarjeta Microcontrolador . . . 35
3.4.3 Salida PWM con Tarjeta Microcontrolador . . . 36
3.4.4 Conversión digital a analógico . . . 38
3.4.5 Comunicaciones en microcontroladores . . . 39
3.5 Actuadores . . . 42
3.6 Sensores . . . 47
3.7 HMI . . . 49

4 Otros dispositivos programables **53**
4.1 La computadora Raspberry Pi . . . 53
4.2 PLD . . . 54
4.3 PLC . . . 54

Bibliografía **56**

Créditos de figuras **57**

Índice de figuras

1.1 Tabla de verdad y símbolo del Buffer. . . . 1
1.2 Tabla de verdad y símbolo de la compuerta NOT. . . . 2
1.3 Tabla de verdad y símbolo de la compuerta OR. . . . 2
1.4 Tabla de verdad y símbolo de la compuerta AND. . . . 2
1.5 Tabla de verdad, símbolo de la compuerta NOR y equivalente. . . . 3
1.6 Tabla de verdad, símbolo de la compuerta NAND y equivalente. . . . 3
1.7 Tabla de verdad, símbolo de la compuerta XOR y equivalente. . . . 4
1.8 Tabla de verdad, símbolo de la compuerta XNOR y equivalente. . . . 4
1.9 Teoremas del 1 al 8 y su representación simbólica. . . . 5
1.10 Diagrama para el ejemplo 1. . . . 5
1.11 Diagrama simplificado para el ejemplo 1. . . . 5
1.12 Diagrama para el ejemplo 2. . . . 6
1.13 Tabla de verdad y análisis del ejemplo 2. . . . 6
1.14 Diagrama simplificado del ejemplo 2. . . . 7
1.15 Ejemplo de tres entradas y agrupamiento en pares. . . . 7
1.16 Ejemplo de agrupamiento cuádruple. . . . 7
1.17 Ejemplo de agrupamiento óctuple. . . . 8
1.18 Ejemplos diversos de uso de mapa “K”. . . . 8
1.19 Mapa “K” del ejemplo 3. . . . 9
1.20 Decodificador de 3 entradas. . . . 10
1.21 Decodificador de binario de 4 bits a 7 segmentos. . . . 10
1.22 Codificador de 8 entradas a binario. . . . 11
1.23 Esquema en bloque de un multiplexor. . . . 11
1.24 Ejemplo de multiplexor de 4 entradas, para sistema de 1 bit. . . . 11
1.25 Esquema en bloque de un demultiplexor. . . . 12

1.26 Ejemplos de suma binaria. . . . 12
1.27 Ejemplo de un sumador de 5 bits. . . . 12
1.28 Tabla de verdad y esquema de un sumador completo. . . . 13
1.29 Sumador completo de un bit. . . . 13
1.30 Tabla de complemento a 2 de 4 bits. . . . 14
1.31 Ejemplo de obtención del complemento a 2. . . . 14
1.32 Un número y su complemento a 2. . . . 14
1.33 Suma de número positivo y negativo usando complemento a 2 de 5 bits. . . . 14
1.34 Flip-Flop SR y su tabla de verdad. . . . 15
1.35 Flip-Flop JK y su tabla de verdad. . . . 15
1.36 Flip-Flop D y su tabla de verdad. . . . 15
1.37 Detector de Flancos. . . . 16
1.38 Algunos contadores de la serie 74ALS16X. . . . 16
1.39 Ejemplo de contador BCD con 74LS90 & 7447. . . . 16
1.40 Registro de 8 bits 74LS374. . . . 17
1.41 Memoria EEPROM 28C256. . . . 18

2.1 Arquitectura Von Neumann. . . . 19
2.2 Arquitectura Harvard. . . . 20
2.3 Ejemplo de ALU 74HC382. . . . 20
2.4 Ejemplo de interconexión de Buses. . . . 21
2.5 Apariencia y descripción de pines del Z80. . . . 21
2.6 Apariencia y descripción de pines del 8086. . . . 22
2.7 Apariencia y descripción de pines del 8088. . . . 23
2.8 Apariencia y descripción de pines del Motorola 68000. . . . 24
2.9 Zócalo LGA2066. . . . 25
2.10 Zócalo sTRX4. . . . 25
2.11 Microprocesador ARM. . . . 26
2.12 Tarjeta Madre. . . . 27

3.1 Distribución de pines y apariencia del PIC16F872. . . . 29
3.2 Distribución de pines y apariencia del ATMEGA328P. . . . 30
3.3 Tarjeta Arduino UNO. . . . 31
3.4 Raspberry Pi Pico. . . . 32
3.5 ESP32. . . . 33
3.6 Diagrama para el programa “Compuerta AND”. . . . 34
3.7 Diagrama para el programa “Condicional IF”. . . . 35
3.8 Diagrama para establecer una entrada analógica. . . . 36
3.9 Ejemplos de PWM con diferentes ciclos de trabajo. . . . 37
3.10 Diagrama para el ejemplo de salida PWM. . . . 37
3.11 Convertidor R-2R. . . . 38
3.12 Conexión de DAC con Arduino. . . . 38
3.13 Forma de onda esperada en la salida del ejemplo DAC. . . . 39
3.14 Diagrama funcional del ejemplo UART. . . . 40
3.15 Diagrama para el ejemplo de comunicación I2C. . . . 41
3.16 Diagrama para el ejemplo de comunicación Bluetooth. . . . 41
3.17 Arduino UNO conectado a Ethernet Shield. . . . 42
3.18 Tarjeta de relevadores electromagnéticos (+5 V). . . . 43
3.19 Relevador de Estado Sólido (SSR). . . . 43
3.20 Motor CC de imán permanente. . . . 44
3.21 Motor paso a paso monopolar y bipolar. . . . 44
3.22 Motor Brushless. . . . 45

3.23 Servomotor industrial y servomotor para pequeños proyectos. 45
3.24 Motor de inducción monofásico. 46
3.25 Control de foco 120 V AC con Arduino UNO. 46
3.26 Control de servo motor con Arduino. 47
3.27 Conexión de sensor HC-SR04 con Arduino. 48
3.28 Conexión de sensor LDR con Arduino. 49
3.29 Conexión de display y teclado. 49
3.30 Formulario del ejemplo de HMI Visual Studio. 51

4.1 Puerto GPIO de Raspberry Pi. 53
4.2 Tarjeta entrenador de FPGA. 54
4.3 PLC. 54

Capítulo 1

Lógica digital

La lógica digital es una rama de la electrónica que se ocupa del diseño y la aplicación de circuitos electrónicos para procesar y controlar información digital. En la lógica digital, las señales se representan mediante dos niveles de voltaje distintos, que corresponden a los dos estados posibles de un bit: 0 (cero) y 1 (uno). Estos estados representan la información binaria fundamental que se utiliza en los sistemas de computación y comunicaciones digitales.

1.1 Lógica combinacional

La lógica combinacional, también conocida como lógica combinatoria, es un tipo de lógica digital donde las salidas de un circuito o sistema se determinan exclusivamente por las combinaciones actuales de sus entradas, sin considerar el estado anterior de las mismas.

1.1.1 Compuertas lógicas

Las compuertas lógicas son los elementos básicos de los circuitos digitales. Son dispositivos que operan con valores binarios (0 y 1, a menudo interpretados como falso y verdadero, respectivamente) y se utilizan para realizar operaciones lógicas básicas.

Buffer

Un buffer, en el contexto de la lógica digital, es un circuito que tiene la función de transferir una entrada a una salida con el objetivo de aumentar la capacidad de manejo de la señal o de aislar diferentes partes de un circuito.

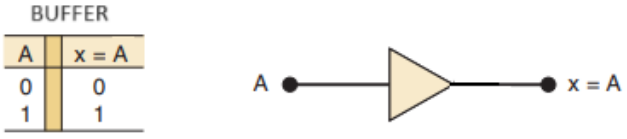


Figura 1.1: Tabla de verdad y símbolo del Buffer.

NOT

La salida será el estado lógico contrario al de la entrada.

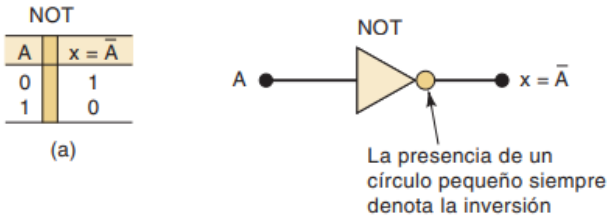


Figura 1.2: Tabla de verdad y símbolo de la compuerta NOT.

OR

Puede tener muchas entradas, pero solo una salida. La salida será falsa solo si todas las entradas son falsas; en caso contrario, la salida será verdadera.

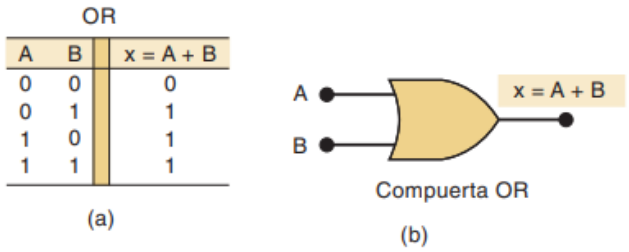


Figura 1.3: Tabla de verdad y símbolo de la compuerta OR.

AND

Puede tener muchas entradas, pero solo una salida. La salida será falsa siempre que exista una o más entradas falsas; en caso contrario, será verdadera.

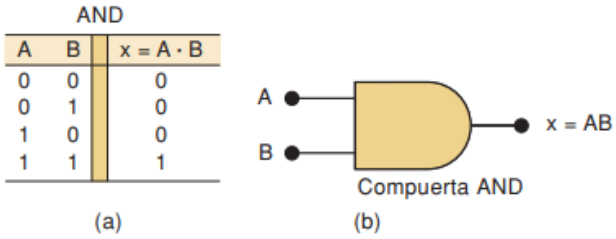


Figura 1.4: Tabla de verdad y símbolo de la compuerta AND.

NOR

Puede tener muchas entradas, pero solo una salida. Es la negación de la OR. La salida será verdadera solo si todas las entradas son falsas; en caso contrario, la salida será falsa.

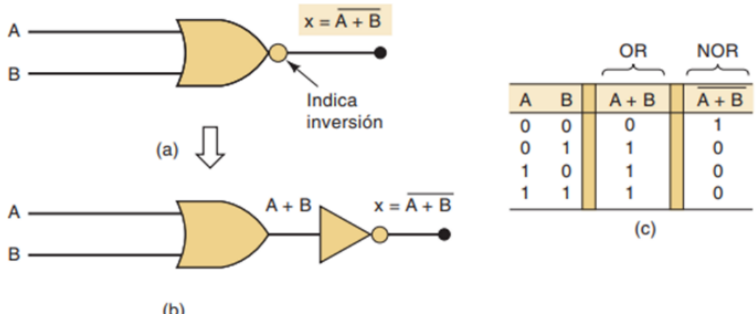


Figura 1.5: Tabla de verdad, símbolo de la compuerta NOR y equivalente.

NAND

Puede tener muchas entradas, pero solo una salida. Es la negación de la AND. La salida será verdadera siempre que exista una o más entradas falsas; en caso contrario, será falsa.

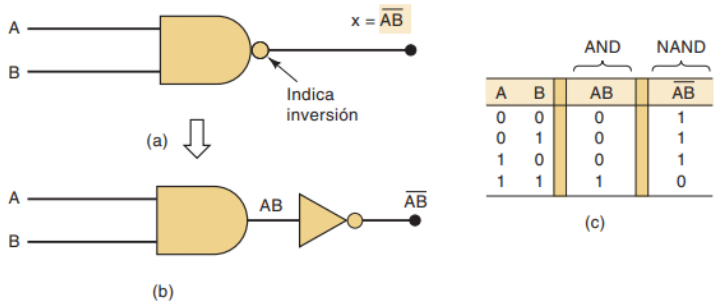


Figura 1.6: Tabla de verdad, símbolo de la compuerta NAND y equivalente.

XOR

Puede tener muchas entradas, pero solo una salida. Es una OR exclusiva. La salida será falsa siempre que todas las entradas tengan el mismo estado lógico; en caso contrario, será verdadera.

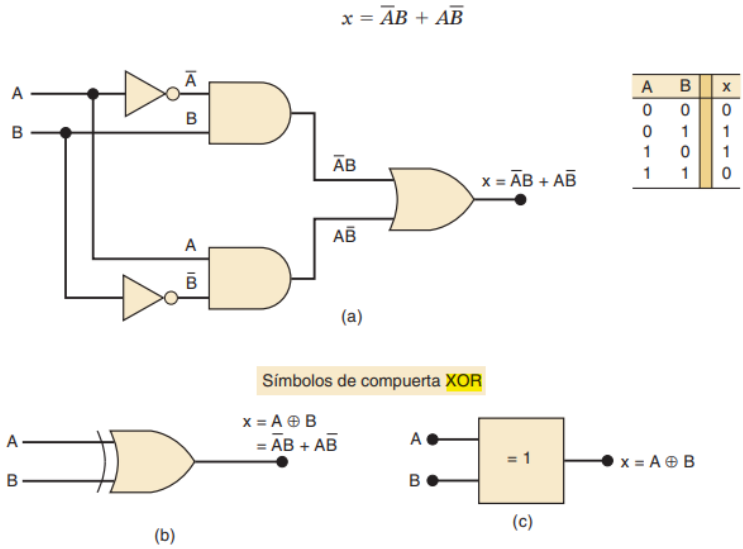


Figura 1.7: Tabla de verdad, símbolo de la compuerta XOR y equivalente.

XNOR

Puede tener muchas entradas, pero solo una salida. Es una OR exclusiva negada. La salida será verdadera siempre que todas las entradas tengan el mismo estado lógico; en caso contrario, será falsa.

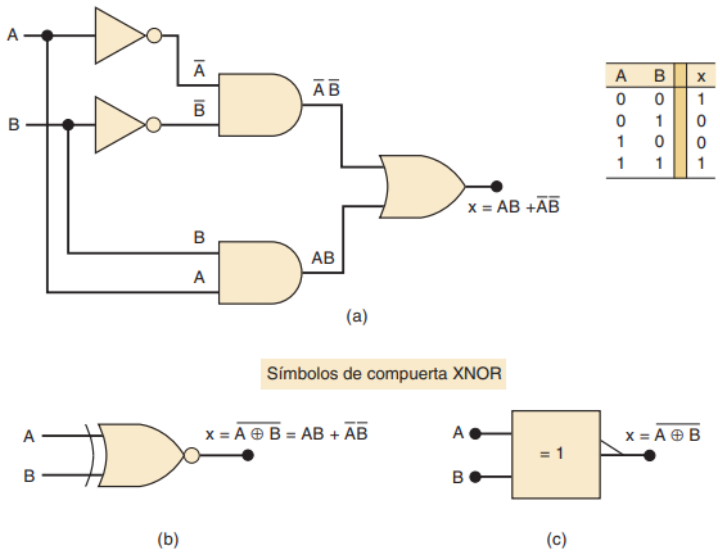


Figura 1.8: Tabla de verdad, símbolo de la compuerta XNOR y equivalente.

1.1.2 Teoremas del Álgebra Booleana

Los teoremas booleanos son principios fundamentales en el álgebra de Boole, que es un área de la matemática dedicada al estudio de las operaciones y las expresiones que involucran valores de verdad binarios (verdadero o falso, representados generalmente como 1 o 0). Estos teoremas proporcionan las reglas básicas para simplificar expresiones booleanas, las cuales son esenciales en el diseño y análisis de circuitos lógicos y sistemas digitales.

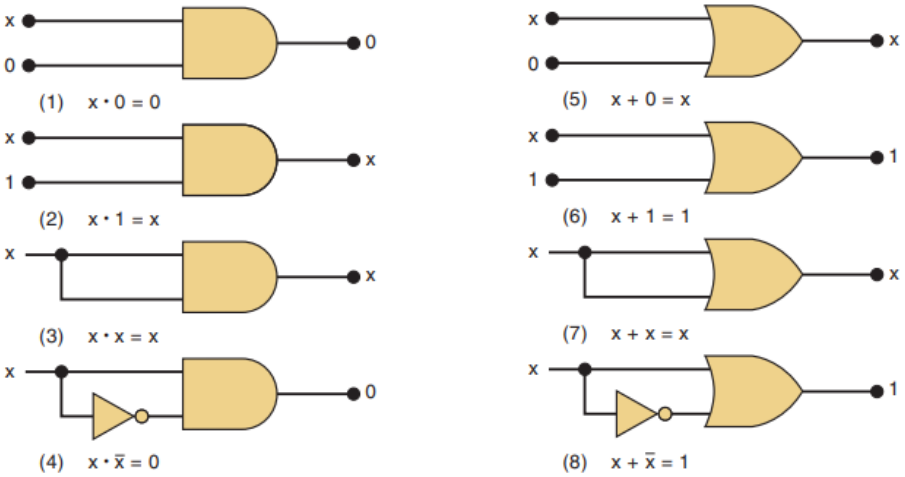


Figura 1.9: Teoremas del 1 al 8 y su representación simbólica.

Ejemplo 1 de simplificación con Álgebra Booleana

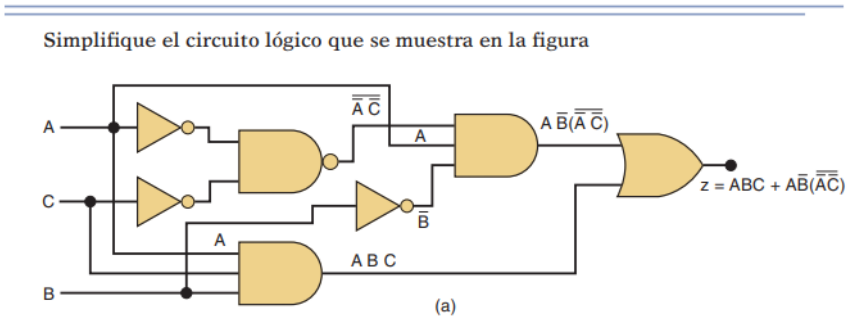


Figura 1.10: Diagrama para el ejemplo 1.

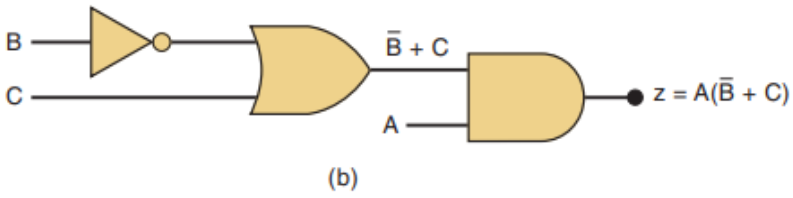


Figura 1.11: Diagrama simplificado para el ejemplo 1.

Ejemplo 2 de simplificación con Álgebra Booleana

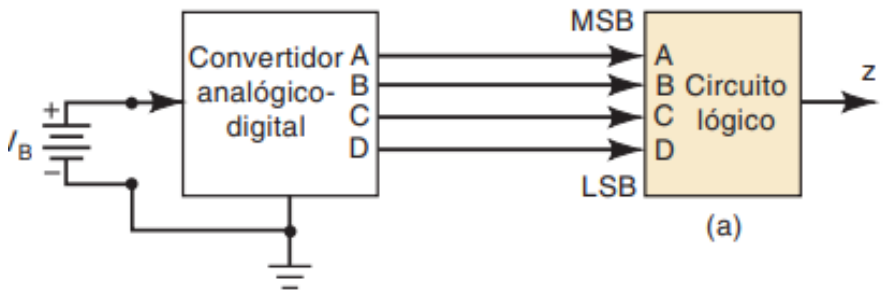


Figura 1.12: Diagrama para el ejemplo 2.

	A	B	C	D	z
(0)	0	0	0	0	0
(1)	0	0	0	1	0
(2)	0	0	1	0	0
(3)	0	0	1	1	0
(4)	0	1	0	0	0
(5)	0	1	0	1	0
(6)	0	1	1	0	0
(7)	0	1	1	1	$1 \rightarrow \bar{A}BCD$
(8)	1	0	0	0	$1 \rightarrow A\bar{B}\bar{C}\bar{D}$
(9)	1	0	0	1	$1 \rightarrow A\bar{B}\bar{C}D$
(10)	1	0	1	0	$1 \rightarrow A\bar{B}C\bar{D}$
(11)	1	0	1	1	$1 \rightarrow A\bar{B}CD$
(12)	1	1	0	0	$1 \rightarrow AB\bar{C}\bar{D}$
(13)	1	1	0	1	$1 \rightarrow AB\bar{C}D$
(14)	1	1	1	0	$1 \rightarrow ABC\bar{D}$
(15)	1	1	1	1	$1 \rightarrow ABCD$

(b)

Figura 1.13: Tabla de verdad y análisis del ejemplo 2.

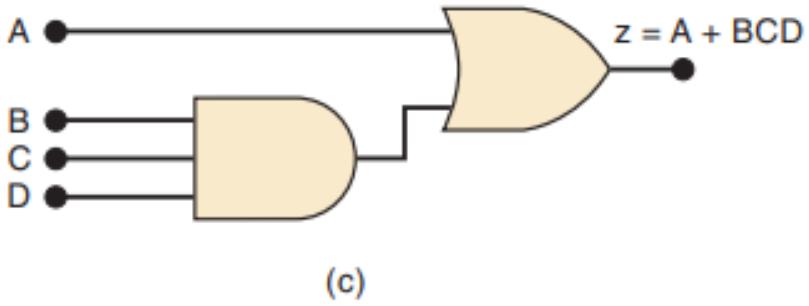


Figura 1.14: Diagrama simplificado del ejemplo 2.

1.1.3 Mapa de Karnaugh

El mapa de Karnaugh es una herramienta gráfica utilizada para simplificar expresiones booleanas. Permite reducir la complejidad de las funciones lógicas con el objetivo de facilitar el diseño de circuitos digitales más eficientes y económicos. Los mapas de Karnaugh presentan una manera visual de organizar y minimizar las ecuaciones booleanas sin necesidad de recurrir a cálculos algebraicos complejos.

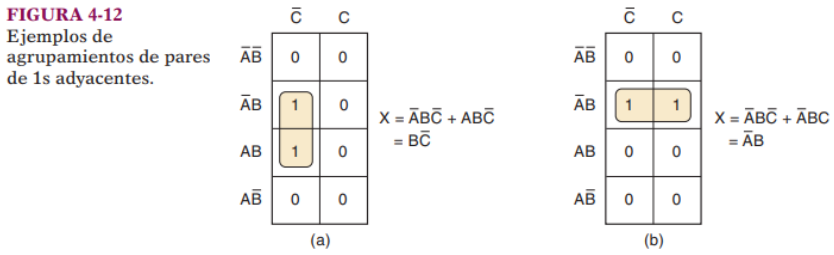


Figura 1.15: Ejemplo de tres entradas y agrupamiento en pares.

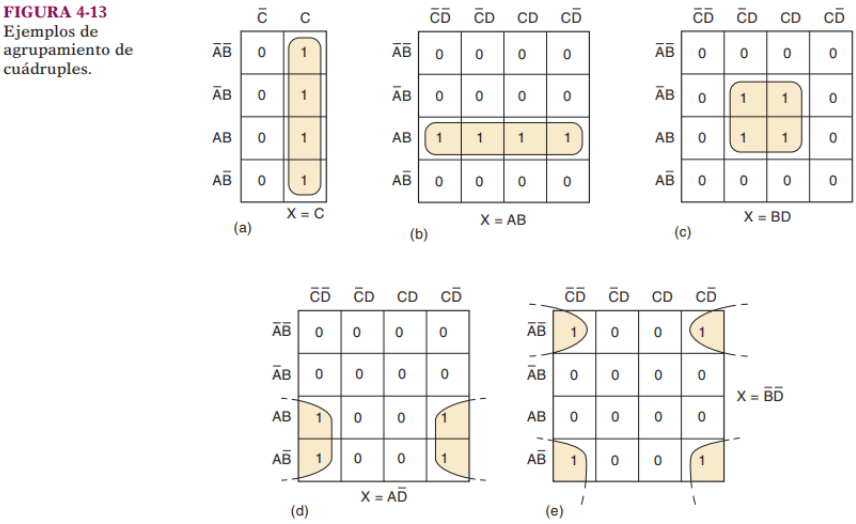


Figura 1.16: Ejemplo de agrupamiento cuádruple.

FIGURA 4-14
Ejemplos de agrupamiento de octetos.

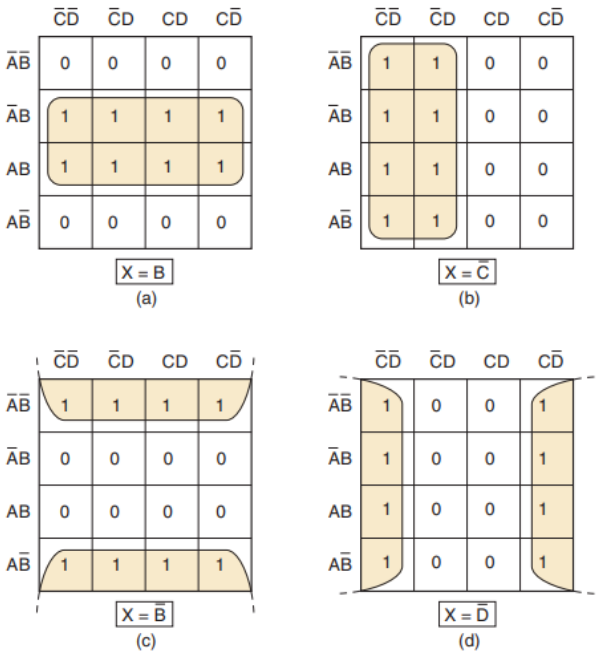


Figura 1.17: Ejemplo de agrupamiento óctuple.

FIGURA 4-15
Ejemplos 4-10 al 4-12.

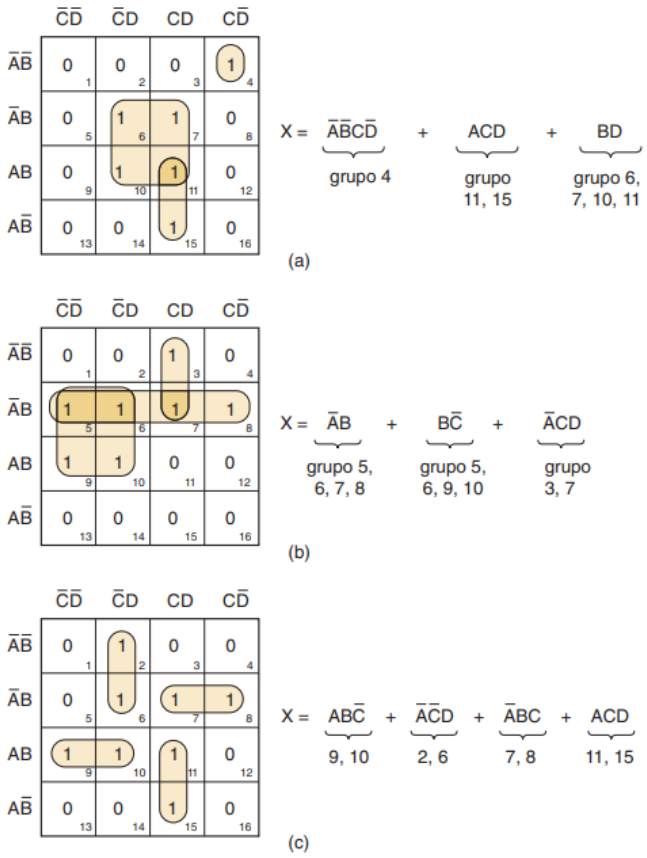


Figura 1.18: Ejemplos diversos de uso de mapa “K”.

Ejemplo 3 (mapa de Karnaugh)

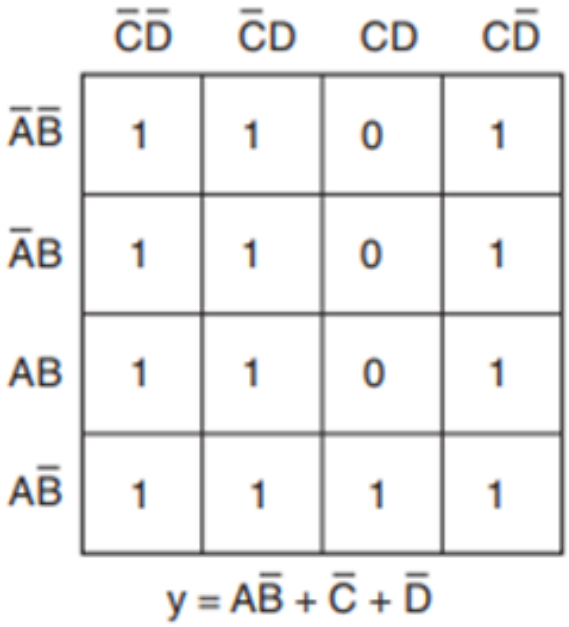


Figura 1.19: Mapa "K" del ejemplo 3.

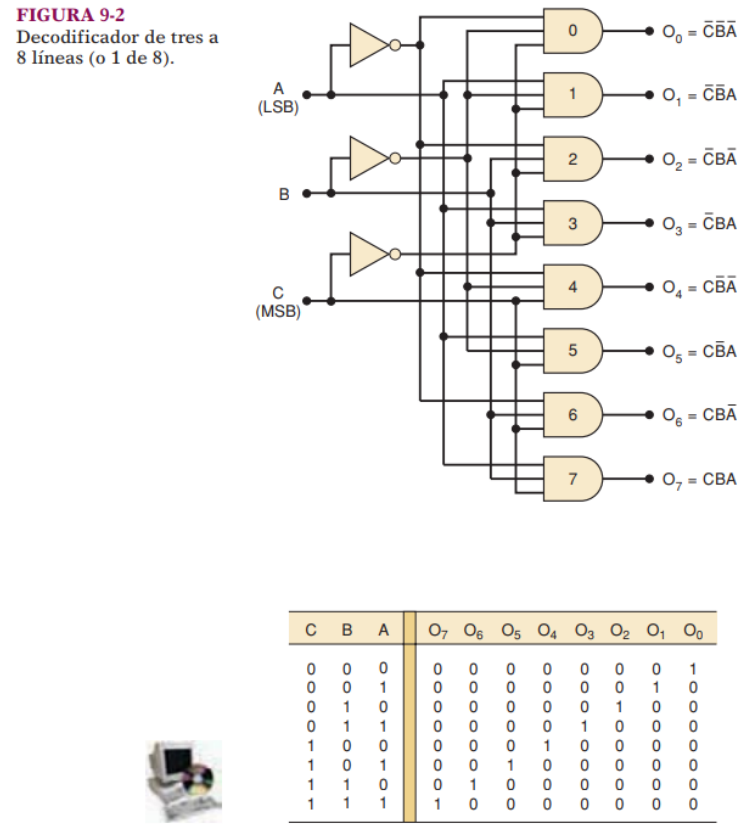
1.1.4 Circuitos MSI

En el contexto de la lógica digital, MSI son las siglas en inglés de *Medium Scale Integration* o Integración a Escala Media. Los circuitos MSI son aquellos que contienen cientos de transistores en un solo chip, permitiendo la implementación de funciones lógicas más complejas que las posibles con circuitos de integración a pequeña escala (SSI), pero sin alcanzar la complejidad de los circuitos de integración a gran escala (LSI) o de muy gran escala (VLSI).

Los circuitos MSI típicamente incorporan de 10 a 100 compuertas lógicas, aunque estas cifras pueden variar según la definición específica.

Decodificadores

Un decodificador es un circuito lógico que acepta un conjunto de entradas que representan un número binario y activa sólo la salida que corresponde a ese número de entrada.



C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Figura 1.20: Decodificador de 3 entradas.

Decodificadores de BCD a 7 Segmentos

Un decodificador de BCD a 7 segmentos es un circuito digital que convierte una entrada de código binario decimal (BCD) en una salida que controla un display de 7 segmentos para mostrar los dígitos decimales del 0 al 9 de forma visual.

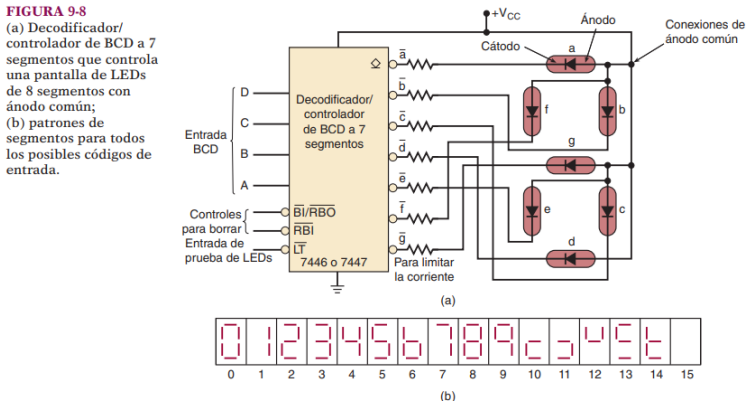


Figura 1.21: Decodificador de binario de 4 bits a 7 segmentos.

Codificadores

Al proceso opuesto a la decodificación se le conoce como codificación y se lleva a cabo mediante un circuito lógico llamado codificador, el cual tiene cierto número de líneas de entrada, de

las cuales sólo una se activa en un momento dado, y produce un código de salida de N bits dependiendo de la entrada que se active.

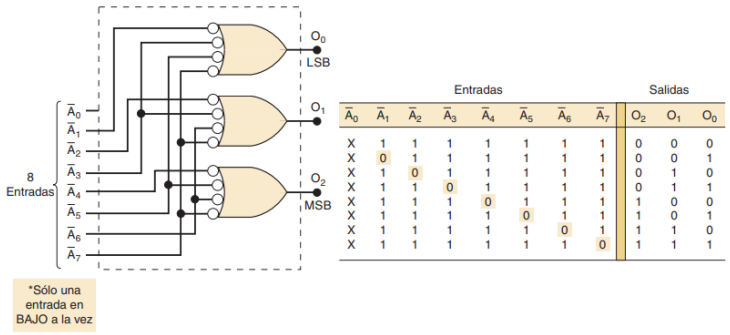


FIGURA 9-13 Circuito lógico para un codificador de octal a binario (de 8 a 3 líneas). Para una operación apropiada, sólo debe haber una entrada activa en un momento dado.

Figura 1.22: Codificador de 8 entradas a binario.

Multiplexores y Demultiplexores

Un multiplexor (mux) es un dispositivo en la lógica digital que selecciona una de las varias entradas analógicas o digitales y la reenvía a la salida única. Funciona como un interruptor múltiple, controlado por señales de selección que determinan cuál de las entradas se conecta a la salida.

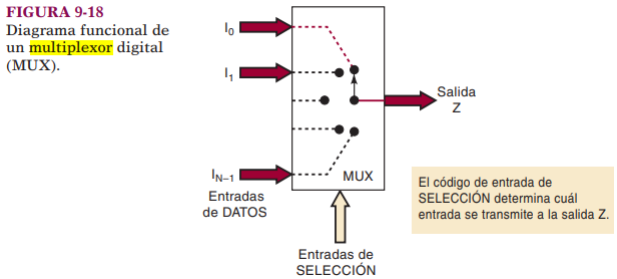


FIGURA 9-18 Diagrama funcional de un multiplexor digital (MUX).

Figura 1.23: Esquema en bloque de un multiplexor.

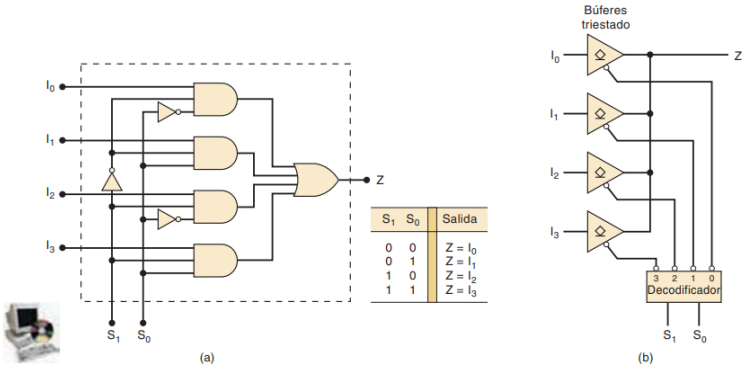


FIGURA 9-20 Multiplexor de cuatro entradas; (a) uso de la lógica de suma de productos; (b) uso de búferes triestado.

Figura 1.24: Ejemplo de multiplexor de 4 entradas, para sistema de 1 bit.

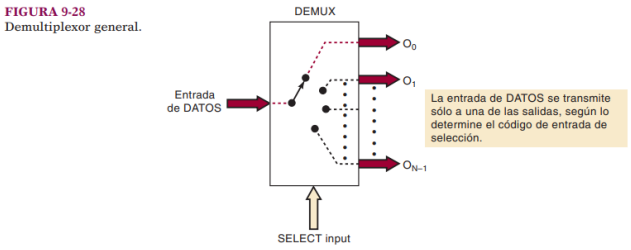


Figura 1.25: Esquema en bloque de un demultiplexor.

1.1.5 Circuitos Aritméticos

Los circuitos aritméticos en el contexto de la lógica digital son circuitos electrónicos diseñados para realizar operaciones matemáticas básicas, como la suma, resta, multiplicación y división, así como operaciones más complejas, utilizando señales digitales.

Sumadores

Un sumador en sistemas digitales es un circuito lógico que se utiliza para realizar la suma de dos números. Los tipos más básicos y comunes son el *half-adder* (medio sumador) y el *full-adder* (sumador completo).

011 (3)	1001 (9)	11.011 (3.375)
+ 110 (6)	+ 1111 (15)	+ 10.110 (2.750)
1001 (9)	11000 (24)	110.001 (6.125)

Figura 1.26: Ejemplos de suma binaria.

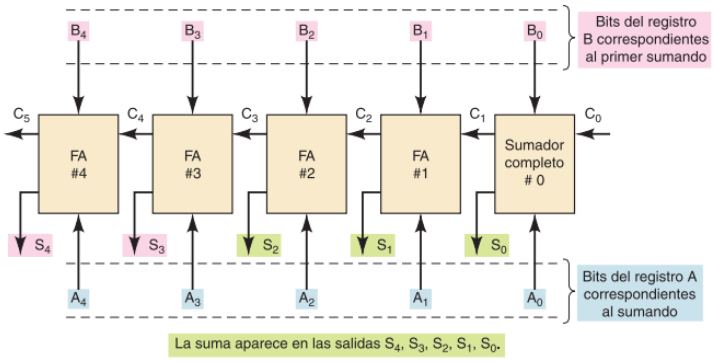


FIGURA 6-6 Diagrama de bloques de un circuito sumador en paralelo de cinco bits, mediante el uso de sumadores completos.

Figura 1.27: Ejemplo de un sumador de 5 bits.

FIGURA 6-7 Tabla de verdad para un circuito sumador completo.

Bit de entrada del primer sumando	Bit de entrada del sumando	Bit de entrada del acarreo	Bit de salida de la suma	Bit de salida del acarreo
A	B	C_{ENT}	S	C_{SAL}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

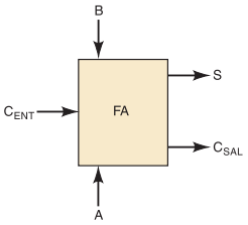


Figura 1.28: Tabla de verdad y esquema de un sumador completo.

FIGURA 6-8 Circuitos completos para un sumador completo.

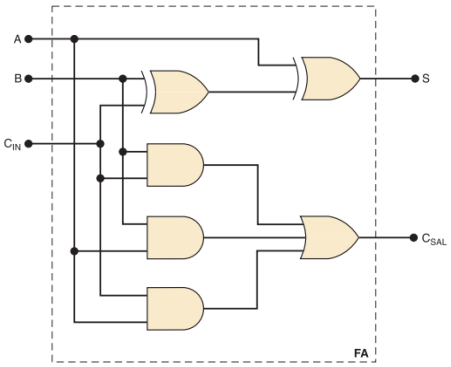


Figura 1.29: Sumador completo de un bit.

Complemento a 2

El complemento a dos es un método utilizado en la computación para representar números enteros positivos y negativos en sistemas binarios. En el complemento a dos, el bit más significativo se utiliza como bit de signo: 0 para números positivos y 1 para números negativos.

Para obtener el complemento a dos de un número binario se sigue este proceso:

1. Invertir todos los bits del número (complemento a uno).
2. Sumar 1 al resultado del paso anterior.

Valor decimal	Binario con signo mediante el uso del complemento a 2
$+7 = 2^3 - 1$	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
$-8 = -2^3$	1000

Figura 1.30: Tabla de complemento a 2 de 4 bits.

```
  1 0 1 1 0 1    equivalente binario de 45
  0 1 0 0 1 0    se complementa cada bit para formar el complemento a 1
+         1      se le suma 1 para formar el complemento a 2
  0 1 0 0 1 1    complemento a 2 del número binario original
```

Figura 1.31: Ejemplo de obtención del complemento a 2.

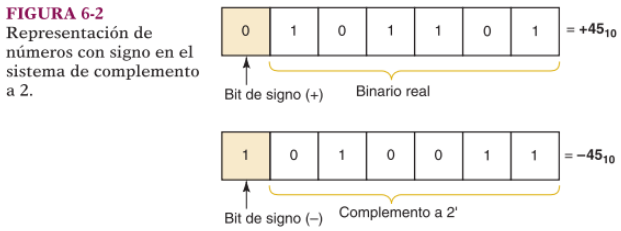


Figura 1.32: Un número y su complemento a 2.

```
  01001   (+9)
+ 11100   (−4)
1 00101   (+5)
↑
└ Se descarta, por lo que el resultado es 00101 = +5.
```

Figura 1.33: Suma de número positivo y negativo usando complemento a 2 de 5 bits.

1.2 Lógica secuencial

En el contexto de lógica digital, la lógica secuencial se refiere a un tipo de diseño de circuitos lógicos en el cual los valores de salida dependen no solo de las combinaciones actuales de valores de entrada, sino también de la historia de los valores de entrada. Esto significa que la lógica secuencial tiene una noción de memoria o estado.

Los componentes clave de la lógica secuencial incluyen *flip-flops* y *latches*, que son circuitos capaces de almacenar un bit de información.

1.2.1 Flip-Flop

Un flip-flop es un circuito electrónico que tiene dos estados estables y se utiliza para almacenar información binaria. Es el bloque de construcción básico de la memoria en los sistemas de lógica digital y se utiliza ampliamente en el diseño de circuitos secuenciales.

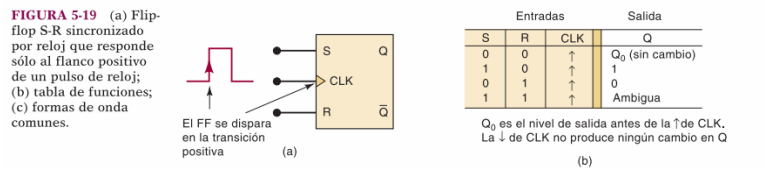


Figura 1.34: Flip-Flop SR y su tabla de verdad.

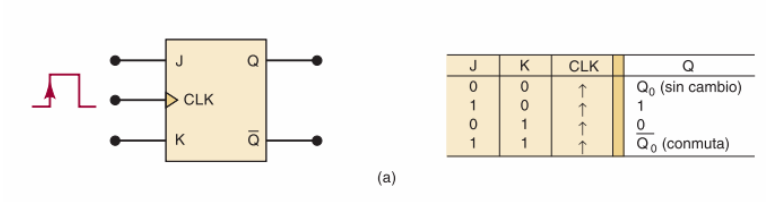


Figura 1.35: Flip-Flop JK y su tabla de verdad.

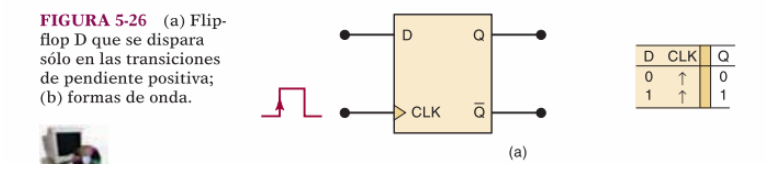


Figura 1.36: Flip-Flop D y su tabla de verdad.

1.2.2 Detector de flancos

Un detector de flancos es un circuito o dispositivo electrónico diseñado para identificar y reaccionar ante cambios repentinos (flancos) en una señal eléctrica. Esencialmente, detecta la transición de una señal de un nivel bajo a un nivel alto (flanco ascendente) o de un nivel alto a un nivel bajo (flanco descendente).

Estos dispositivos se utilizan en una amplia gama de aplicaciones, incluidos los sistemas de comunicaciones, controles digitales, interruptores electrónicos y sistemas embebidos.

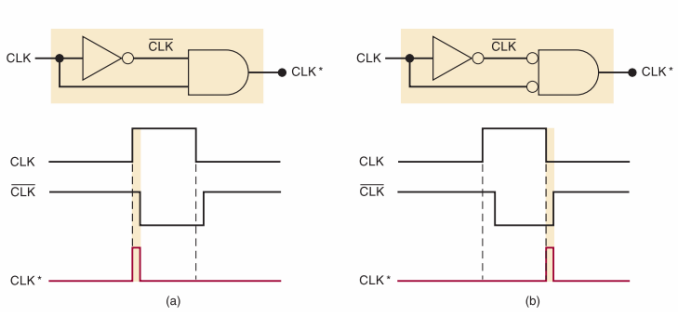


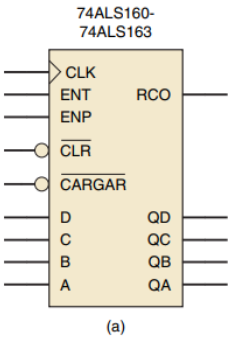
FIGURA 5-22 Implementación de circuitos detectores de flancos que se utilizan en flip-flops disparados por flanco: (a) PGT; (b) NGT. La duración de los pulsos en CLK* es por lo general de 2 a 5 ns.

Figura 1.37: Detector de Flancos.

1.2.3 Contador binario

Un contador binario es un tipo de circuito secuencial digital que pasa por una secuencia predefinida de estados binarios en respuesta a una secuencia de pulsos de entrada, generalmente provistos por un reloj.

FIGURA 7-13
Contadores síncronos de la serie 74ALS160-74ALS163: (a) símbolo lógico, (b) módulos; (c) tabla de funciones.



Número de pieza	Módulo
74ALS160	10
74ALS161	16
74ALS162	10
74ALS163	16

(b)

Figura 1.38: Algunos contadores de la serie 74ALS16X.

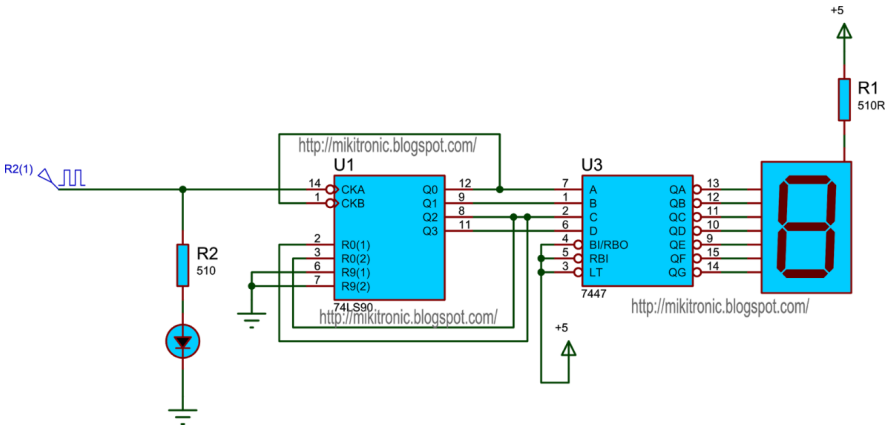


Figura 1.39: Ejemplo de contador BCD con 74LS90 & 7447.

1.2.4 Registros

En el contexto de la electrónica digital, un registro es un dispositivo o circuito que se utiliza para almacenar temporalmente una palabra de datos binarios. Está compuesto por una serie de flip-flops, cada uno capaz de almacenar un bit de información.

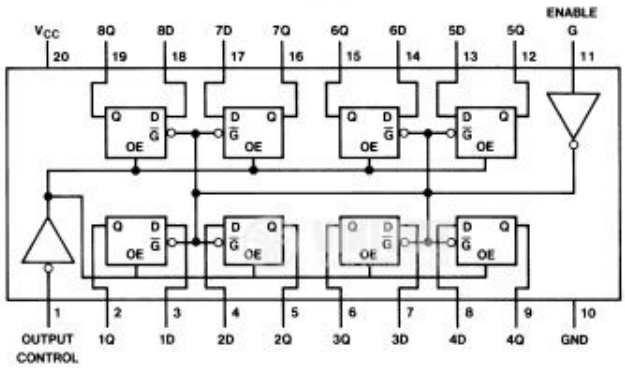


Figura 1.40: Registro de 8 bits 74LS374.

1.2.5 Memorias

En el contexto de la electrónica digital, la memoria se refiere a los dispositivos o sistemas que se utilizan para almacenar datos o instrucciones de manera temporal o permanente.

Las memorias en la electrónica digital se pueden clasificar en varios tipos:

- **Memoria Volátil:** Su contenido se pierde al apagar el dispositivo. Incluye:
  - **RAM** (Random Access Memory): memoria de trabajo principal.
  - **DRAM** (Dynamic RAM): necesita refresco periódico.
  - **SRAM** (Static RAM): más rápida, no requiere refresco.
- **Memoria No Volátil:** Mantiene la información sin alimentación. Incluye:
  - **ROM** (Read-Only Memory): datos raramente modificados.
  - **EPROM**: borrable con luz ultravioleta y reprogramable.
  - **EEPROM**: borrable y reprogramable eléctricamente.
  - **Flash Memory**: borrado y escritura por bloques; usada en tarjetas SD, USB y SSDs.
- **Memoria de Almacenamiento Masivo:** Discos duros, SSDs, cintas.
- **Memoria Caché:** Alta velocidad, almacena datos de acceso frecuente; presente en procesadores y sistemas de almacenamiento.

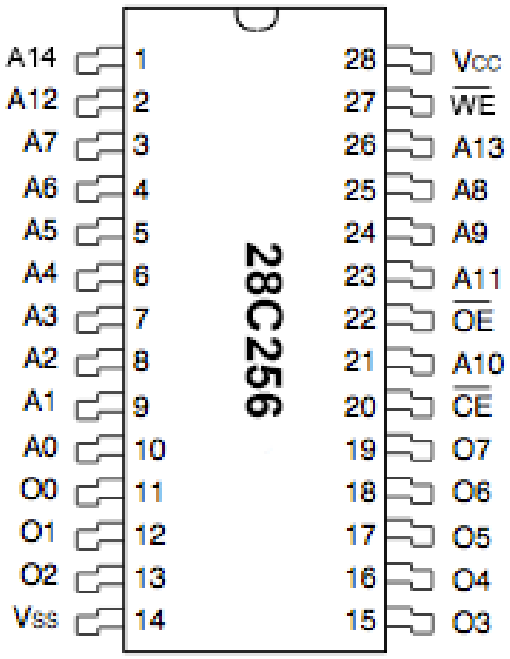


Figura 1.41: Memoria EEPROM 28C256.

Capítulo 2

Microprocesadores

En este capítulo se ofrece una oportunidad fascinante para adentrarse en el corazón de la tecnología moderna. Estos pequeños pero poderosos dispositivos son el núcleo pulsante de las computadoras, responsables de ejecutar las instrucciones que permiten a los dispositivos realizar una amplia gama de funciones.

2.1 Arquitecturas de computadoras

Los modelos de computadoras de Von Neumann y Harvard se destacan por sus enfoques distintivos en el diseño de la arquitectura de sistemas computacionales. El modelo de Von Neumann integra la unidad de procesamiento y la memoria en un espacio compartido, facilitando la ejecución secuencial de instrucciones. El modelo Harvard separa físicamente la memoria de instrucciones de la memoria de datos, permitiendo el acceso simultáneo a ambas y, por ende, una mayor velocidad de procesamiento.

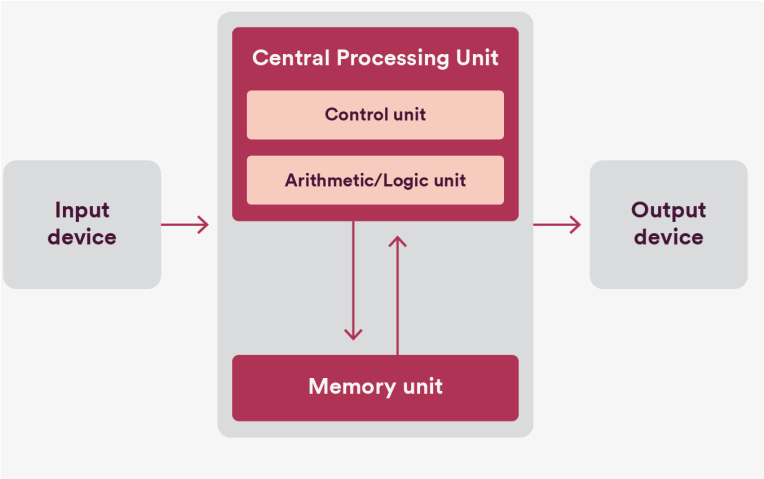


Figura 2.1: Arquitectura Von Neumann.

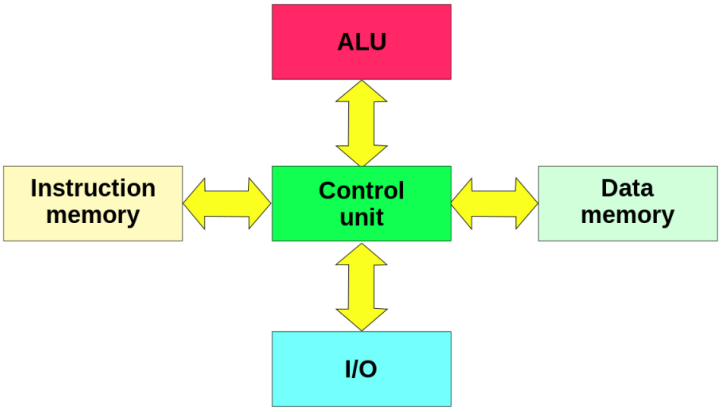


Figura 2.2: Arquitectura Harvard.

2.2 ALU

La Unidad Aritmética y Lógica (ALU, por sus siglas en inglés) es un componente crítico de la CPU. Es responsable de llevar a cabo operaciones aritméticas (suma, resta, multiplicación y división) y operaciones lógicas (AND, OR, NOT y XOR) sobre los datos binarios que procesa el sistema.

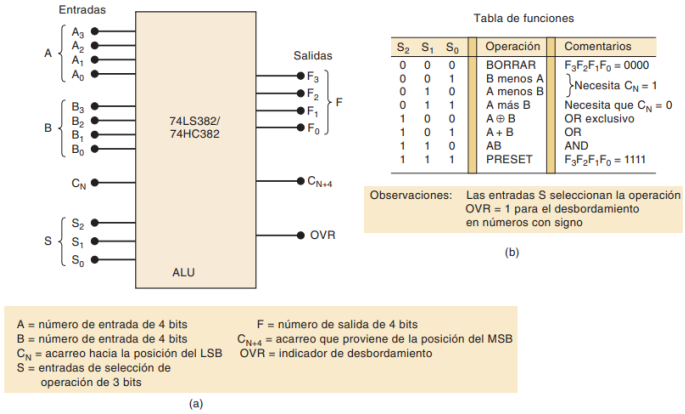


FIGURA 6-15 (a) Símbolo de bloque para el chip ALU 74LS382/HC382; (b) tabla de funciones que muestra cómo las entradas de selección (*S*) determinan la operación que se va a realizar sobre las entradas *A* y *B*.

Figura 2.3: Ejemplo de ALU 74HC382.

2.3 Buses

En el contexto de la electrónica digital, un bus se refiere a un sistema de transmisión que transporta datos, direcciones, señales de control y energía entre los diversos componentes de un dispositivo electrónico.

Hay varios tipos de buses:

- **Bus de Datos:** Transporta los datos entre los componentes del sistema. Su ancho determina la velocidad y rendimiento.
- **Bus de Direcciones:** Lleva las señales de direcciones que especifican las ubicaciones de

memoria o dispositivos de E/S.

- **Bus de Control:** Transporta señales de control como lectura/escritura, reloj e interrupciones.

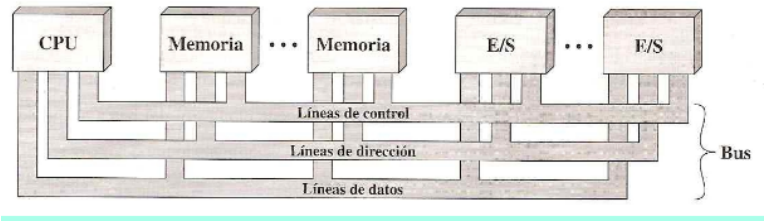


Figura 2.4: Ejemplo de interconexión de Buses.

2.4 Microprocesador

Un microprocesador es un circuito integrado que contiene las funciones de una unidad central de procesamiento (CPU) de una computadora en un único chip compacto. Los componentes principales de un microprocesador incluyen:

- **Unidad Aritmético-Lógica (ALU):** Realiza operaciones matemáticas y lógicas.
- **Registros:** Almacenamiento temporal dentro del microprocesador.
- **Unidad de Control (CU):** Decodifica instrucciones y controla el flujo de datos.
- **Bus de Datos, Bus de Direcciones y Bus de Control.**

2.5 Ejemplos de Microprocesadores

Zilog Z80

El Zilog Z80 es un microprocesador de 8 bits diseñado y comercializado por Zilog a partir de 1976. Se convirtió en uno de los procesadores más populares de su época, ampliamente utilizado en computadoras personales y sistemas embebidos durante finales de los años 70 y la década de los 80.

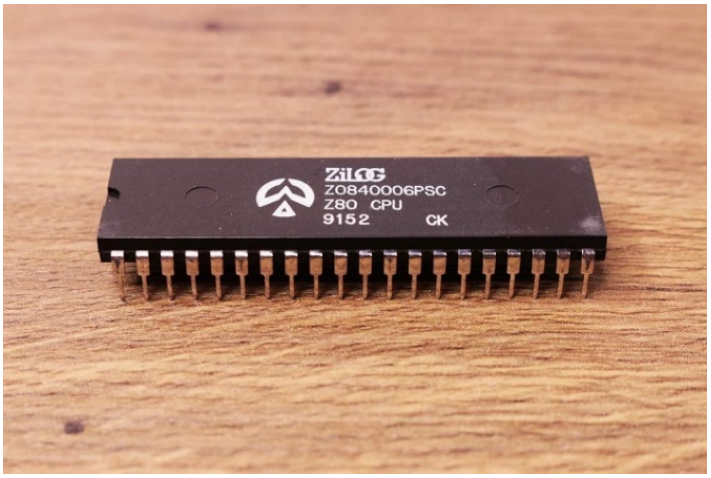


Figura 2.5: Apariencia y descripción de pines del Z80.

Ejemplo de programación del Zilog Z80

```
; Programa para sumar dos numeros en el Zilog Z80
    ORG 0000h       ; Establece la direccion de inicio del programa
    LD A, 19h       ; Carga el primer numero (25) en el registro A
    LD B, 0Fh       ; Carga el segundo numero (15) en el registro B
    ADD A, B        ; Suma el contenido de B al contenido de A
    HALT            ; Detiene la ejecucion del programa
```

Listing 2.1: Suma de dos números en el Zilog Z80.

Intel 8086

El Intel 8086 es un microprocesador de 16 bits introducido por Intel en 1978, marcando el inicio de la arquitectura x86. Características clave:

- **Arquitectura de 16 bits:** procesa datos de 16 bits en una sola operación.
- **Espacio de direccionamiento:** 1 MB mediante esquema de segmentación.
- **Conjunto de instrucciones** rico con múltiples modos de direccionamiento.
- **Compatibilidad** con el software del Intel 8080.

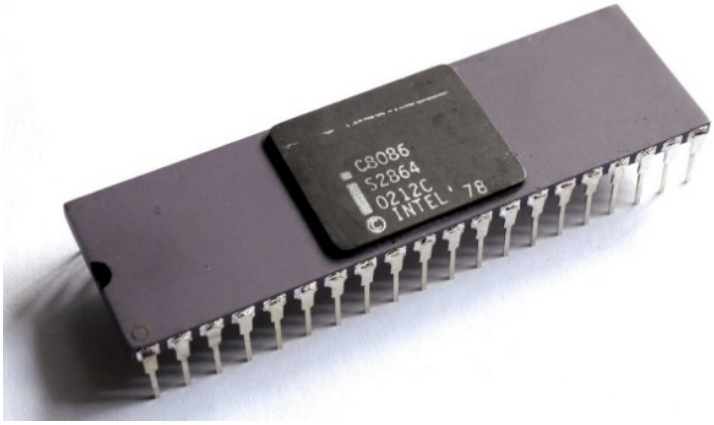


Figura 2.6: Apariencia y descripción de pines del 8086.

Intel 8088

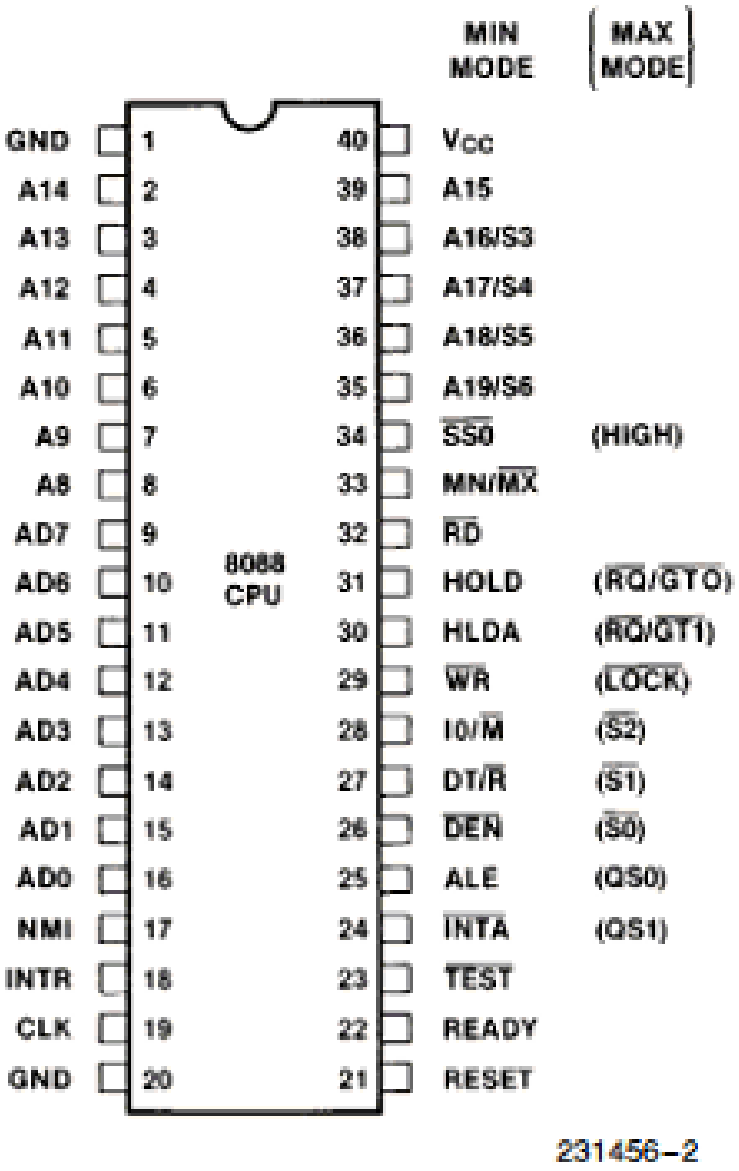


Figura 2.7: Apariencia y descripción de pines del 8088.

Motorola 68000

El Motorola 68000 es un microprocesador de 16/32 bits introducido por Motorola en 1979. Fue utilizado ampliamente en computadoras personales como la Apple Macintosh, la serie Commodore Amiga y Atari ST, y en consolas como la Sega Genesis/Mega Drive.

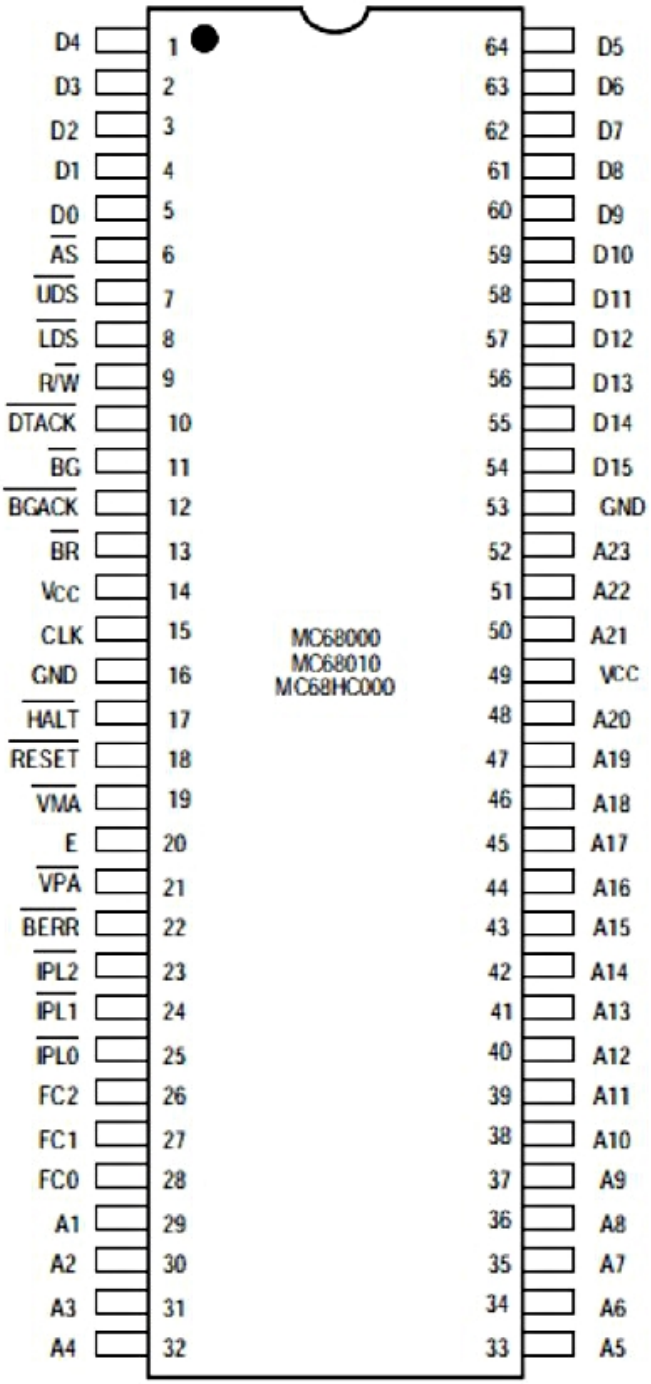


Figura 2.8: Apariencia y descripción de pines del Motorola 68000.

Intel Core i7

El Intel Core i7 es una familia de microprocesadores de alto rendimiento introducida por Intel en 2008. Características clave: múltiples núcleos, Hyper-Threading, Turbo Boost, amplia caché, virtualización y gráficos integrados.

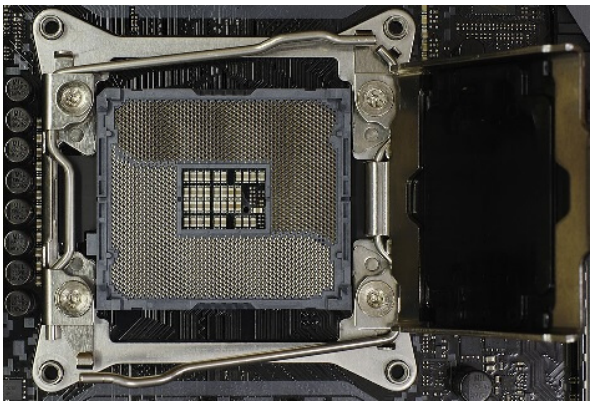


Figura 2.9: Zócalo LGA2066.

AMD Ryzen

Los procesadores AMD Ryzen representan la línea de microprocesadores de alto rendimiento de AMD, lanzada en marzo de 2017. Se basan en la arquitectura "Zen" y han traído significativas mejoras en rendimiento, eficiencia energética y competencia en precio.

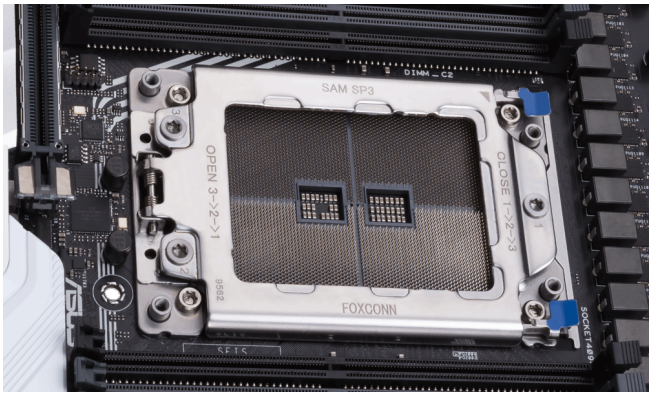


Figura 2.10: Zócalo sTRX4.

ARM

Los microprocesadores ARM son una familia de arquitecturas de CPU basadas en el conjunto de instrucciones RISC, desarrolladas por ARM Holdings. Son conocidos por su eficiencia energética y rendimiento por vatio, lo que los hace particularmente populares en dispositivos móviles y sistemas embebidos.

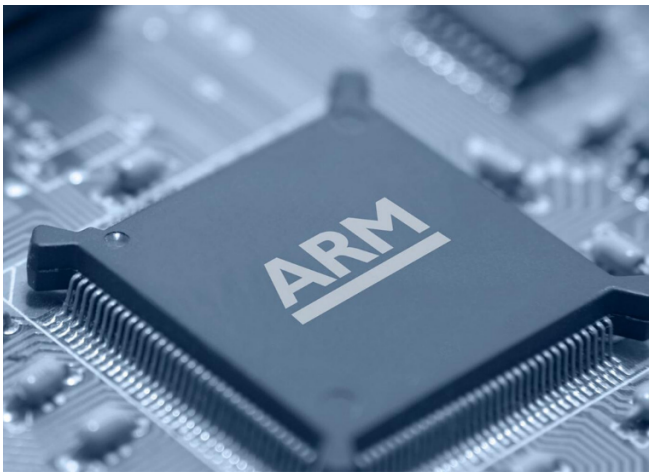


Figura 2.11: Microprocesador ARM.

2.6 Lenguaje de máquina, ensamblador y de alto nivel

Comparar lenguaje de máquina, ensamblador y lenguajes de alto nivel implica observar aspectos como la abstracción, la facilidad de uso, la portabilidad y el control sobre el hardware.

Lenguaje de Máquina

- **Definición:** Instrucciones directamente ejecutables por el procesador, compuestas por bits (0s y 1s).
- **Abstracción:** Ninguna.
- **Portabilidad:** No portátil; depende de la arquitectura.
- **Control sobre el hardware:** Máximo.

Ensamblador

- **Definición:** Usa mnemónicos para representar operaciones de máquina. Cada instrucción se traduce a una instrucción en lenguaje de máquina.
- **Abstracción:** Baja; ligado a la arquitectura del procesador.
- **Portabilidad:** Baja.
- **Control sobre el hardware:** Alto.

Lenguajes de Alto Nivel

- **Definición:** Proporcionan un alto nivel de abstracción del hardware, con conceptos más cercanos al pensamiento humano (p. ej., C++).
- **Abstracción:** Alta.
- **Portabilidad:** Alta; compilables para diferentes plataformas.
- **Control sobre el hardware:** Relativamente limitado, aunque C/C++ permiten un control considerable.

2.7 Tarjeta Madre

Una tarjeta madre (placa base o *motherboard*) es el componente central de una computadora, sobre el cual se montan o conectan los demás componentes. Sus características principales incluyen: zócalo del procesador, ranuras de memoria, chipset, ranuras de expansión (PCIe), conectores de almacenamiento (SATA, M.2), BIOS/UEFI, conectores de E/S y conectores de alimentación.

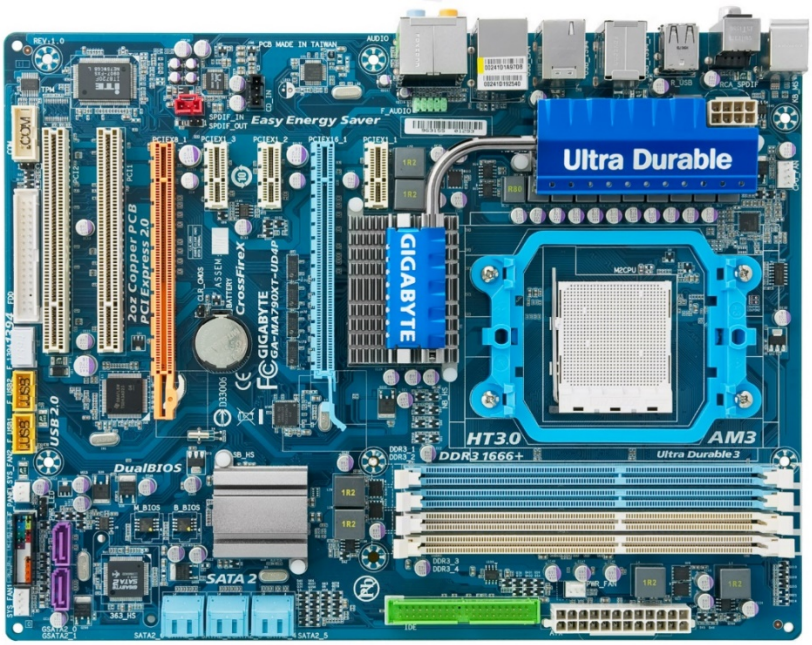


Figura 2.12: Tarjeta Madre.

Capítulo 3

Microcontroladores

Un microcontrolador es un dispositivo compacto de circuito integrado que contiene todos los componentes necesarios para realizar operaciones de control y procesamiento. A diferencia de un microprocesador, un microcontrolador integra un procesador (CPU), memoria (RAM y ROM), dispositivos de E/S y otros periféricos, todo en un solo chip.

Imagina que reduces una tarjeta madre con todos sus componentes instalados a un solo circuito integrado (chip).

3.1 PIC

Los PIC son una familia de microcontroladores desarrollados y comercializados por Microchip Technology Inc. Desde su introducción en la década de 1970, han ganado popularidad gracias a su bajo costo, versatilidad y facilidad de programación. Características clave:

- **Arquitectura RISC:** ejecuta la mayoría de las instrucciones en un solo ciclo de reloj.
- **Variedad de modelos:** desde 8 bits hasta 32 bits.
- **Facilidad de programación:** MPLAB X IDE y compilador XC.
- **Periféricos integrados:** PWM, ADC, USART, SPI, I2C.
- **Bajo consumo de energía:** modos de sueño y operación con bajo voltaje.

Comandos en ensamblador de PIC

Los microcontroladores PIC16FXXX utilizan un conjunto de instrucciones consistente. Principales grupos de instrucciones:

Operaciones Aritméticas ADDWF, SUBWF, INCF, DECF, MULWF

Operaciones Lógicas ANDWF, IORWF, XORWF, COMF, RLF, RRF

Control de Flujo GOTO, CALL, RETURN, RETLW, DECFSZ, INCFSZ

Manipulación de Bit BCF, BSF, BTFSC, BTFSS

Transferencia de Datos MOVWF, MOVF, SWAPF, CLRF, CLRW

Trabajo con Literal MOVLW, IORLW, ANDLW, XORLW

Especiales NOP, SLEEP, CLRWDT

PIC16F872

Figura 3.1: Distribución de pines y apariencia del PIC16F872.

```
    LIST    p=16F872
    #include <p16F872.inc>
    __CONFIG _CONFIG1, _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
       _HS_OSC
    ORG     0x00
    bsf     STATUS, RP0     ; Banco de registros 1
    bcf     TRISA, 0        ; Configura RA0 como salida
    bcf     STATUS, RP0     ; Vuelve al banco 0
MainLoop
    bsf     PORTA, 0        ; Enciende el LED
    call    Delay
    bcf     PORTA, 0        ; Apaga el LED
    call    Delay
    goto    MainLoop
Delay
    movlw   0xFF
DelayLoop
    addlw   -1
    btfss   STATUS, Z
    goto    DelayLoop
    return
    END
```

Listing 3.1: Parpadeo de LED en PIC16F872.

3.2 ATMEGA

ATmega es una familia de microcontroladores desarrollados por Atmel (ahora Microchip Technology), basados en la arquitectura AVR de 8 bits. Son conocidos por su eficiencia energética, capacidades de procesamiento y opciones de memoria integrada, y sirven como plataforma de hardware para el popular Arduino.

ATMEGA328P

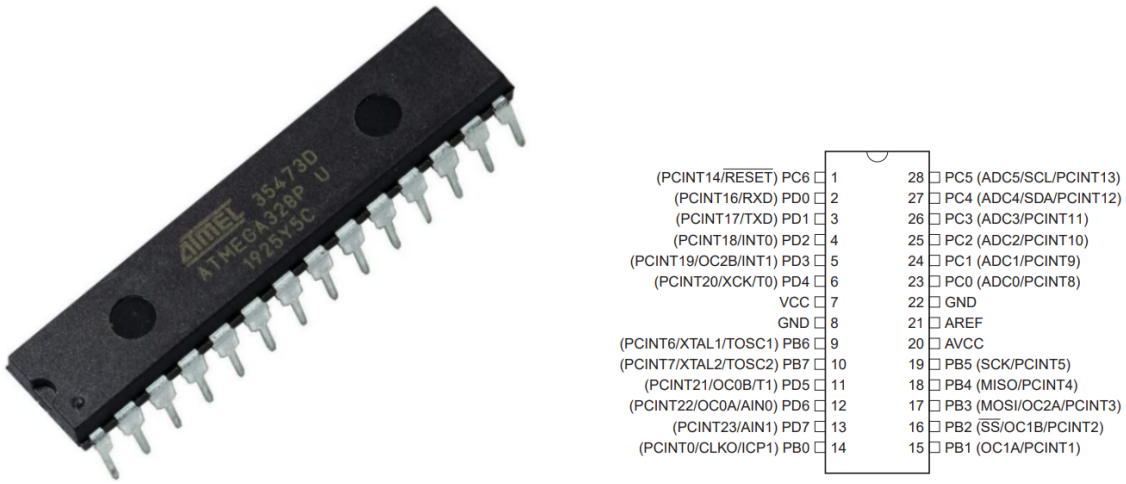


Figura 3.2: Distribución de pines y apariencia del ATMEGA328P.

```
.def    temp = r16
.equ    DDRD = 0x0A
.equ    PORTD = 0x0B
.equ    PD2 = 2

    ldi     temp, (1<<PD2)
    out     DDRD, temp

MainLoop:
    sbi     PORTD, PD2
    call    Delay
    cbi     PORTD, PD2
    call    Delay
    rjmp    MainLoop

Delay:
    ldi     r18, 20
DelayLoop:
    dec     r18
    brne    DelayLoop
    ret
```

Listing 3.2: Parpadeo de LED en ATMEGA328P (ensamblador AVR).

3.3 Comparación entre Microcontrolador y Microprocesador

Cuadro 3.1: Diferencias entre microcontrolador y microprocesador.

Aspecto	Microcontrolador	Microprocesador
Integración	CPU + RAM + ROM + E/S en un chip	Solo CPU; requiere componentes externos
Objetivo	Control de sistemas embebidos	Cómputo general
Aplicaciones	Electrodomésticos, automóviles, IoT	PCs, servidores, estaciones de trabajo
Consumo	Bajo, modos de sueño	Mayor consumo
Costo	Más económico	Puede ser más costoso (sistema completo)
Rendimiento	Adecuado para su aplicación	Alto; ejecuta SO complejos

3.4 Tarjeta Microcontrolador

Una tarjeta microcontrolador es una placa de circuito impreso (PCB) que incorpora un microcontrolador junto con los componentes necesarios para facilitar su programación y utilización en proyectos de electrónica. Características típicas: microcontrolador integrado, regulador de voltaje, interfaces de programación, pines de E/S, LEDs indicadores, botones, cristal oscilador y conectores de expansión.

Arduino UNO

Arduino Uno es una placa de desarrollo basada en el ATmega328P, una de las más populares y accesibles dentro de la familia Arduino.

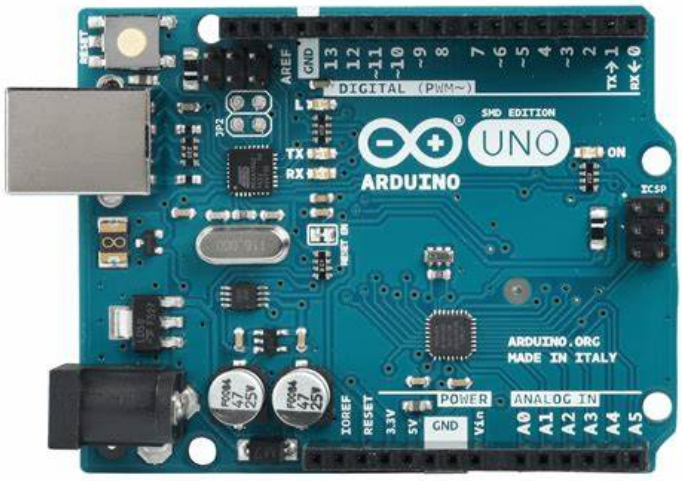


Figura 3.3: Tarjeta Arduino UNO.

Raspberry Pi Pico

La Raspberry Pi Pico es una placa de desarrollo de microcontrolador introducida por la Raspberry Pi Foundation en enero de 2021, basada en el microcontrolador RP2040 con procesador

dual ARM Cortex-M0+ a 133 MHz.

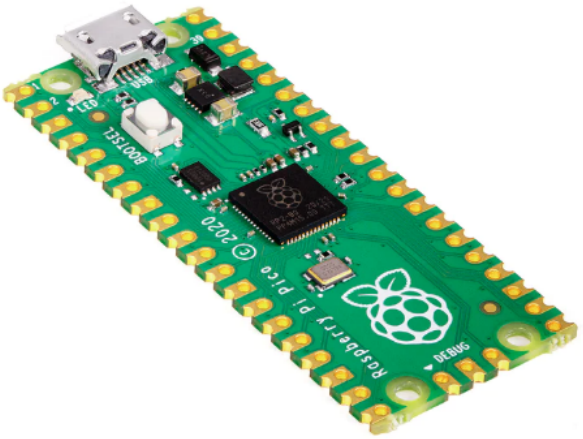


Figura 3.4: Raspberry Pi Pico.

ESP32

El ESP32 es un microcontrolador de bajo costo con conectividad Wi-Fi y Bluetooth integrada, desarrollado por Espressif Systems. Cuenta con procesador Xtensa LX6 de doble núcleo a 240 MHz.

Figura 3.5: ESP32.

3.4.1 Entradas y salidas discretas con Tarjeta Microcontrolador

Las **entradas discretas** reciben señales digitales de dos estados: encendido (1) o apagado (0). Las **salidas discretas** envían señales digitales para controlar dispositivos como LEDs, relés y motores paso a paso.

```
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Listing 3.3: Parpadeo de LED en Arduino UNO.

Compuerta AND con Arduino UNO

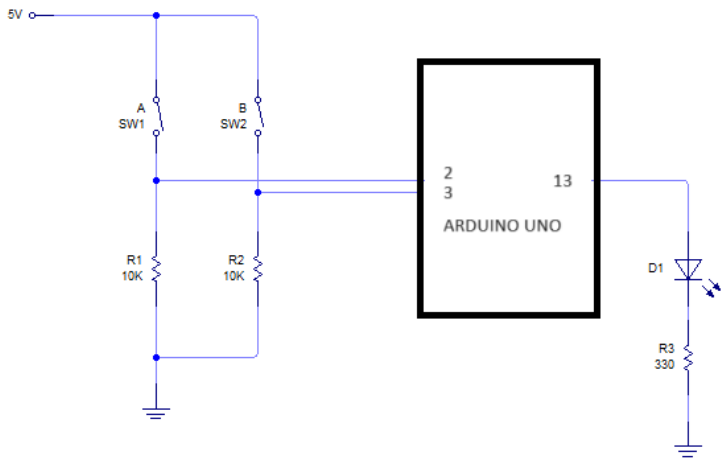


Figura 3.6: Diagrama para el programa “Compuerta AND”.

```
bool A = LOW;
bool B = LOW;
bool Z = LOW;

void setup() {
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(13, OUTPUT);
}

void loop() {
  A = digitalRead(2);
  B = digitalRead(3);
  Z = A && B;
  digitalWrite(13, Z);
}
```

Listing 3.4: Compuerta AND con Arduino UNO.

Condicional IF con Arduino UNO

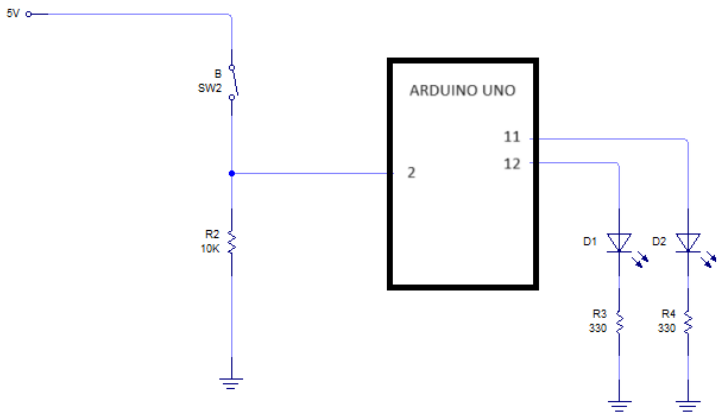


Figura 3.7: Diagrama para el programa “Condicional IF”.

```
bool A = LOW;
bool Z0 = LOW;
bool Z1 = LOW;

void setup() {
  pinMode(2, INPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop() {
  if (digitalRead(2) == HIGH) {
    digitalWrite(11, HIGH);
    digitalWrite(12, LOW);
  }
  if (digitalRead(2) == LOW) {
    digitalWrite(11, LOW);
    digitalWrite(12, HIGH);
  }
}
```

Listing 3.5: Condicional IF con Arduino UNO.

3.4.2 Entrada analógica con Tarjeta Microcontrolador

Las entradas analógicas leen valores que varían de manera continua a lo largo de un rango, capturando señales de termistores, potenciómetros, células de carga, etc. El ADC mapea el voltaje analógico (0–5 V) a un valor digital (0–1023 en 10 bits).

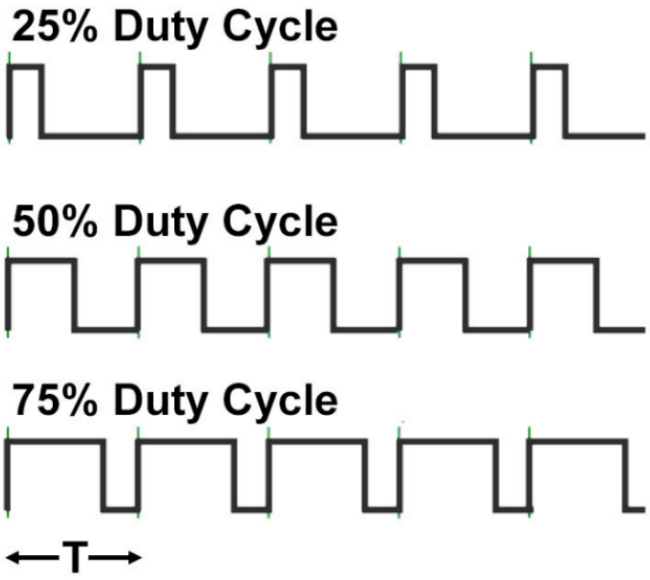


Figura 3.8: Diagrama para establecer una entrada analógica.

```
int val = 0;

void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  val = analogRead(0);
  if (val < 511) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```

Listing 3.6: Entrada analógica simple con Arduino UNO.

3.4.3 Salida PWM con Tarjeta Microcontrolador

PWM (Modulación por Ancho de Pulso) es una técnica para simular una señal analógica mediante la modificación del ancho de los pulsos de una señal digital. El **ciclo de trabajo** (Duty Cycle) se refiere a la fracción del tiempo que la señal está en alto durante un ciclo completo.

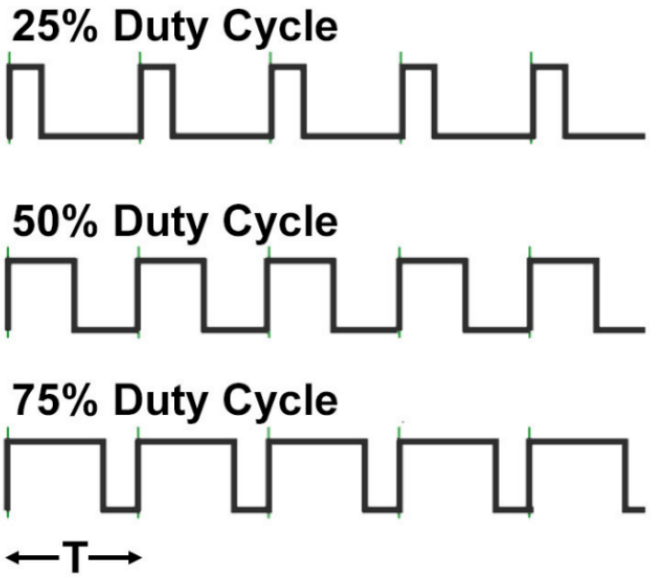


Figura 3.9: Ejemplos de PWM con diferentes ciclos de trabajo.

```
const int ledPin = 9;
const int potPin = A0;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int potValue = analogRead(potPin);
  int pwmValue = map(potValue, 0, 1023, 0, 255);
  analogWrite(ledPin, pwmValue);
  delay(10);
}
```

Listing 3.7: Control de salida PWM con entrada analógica.

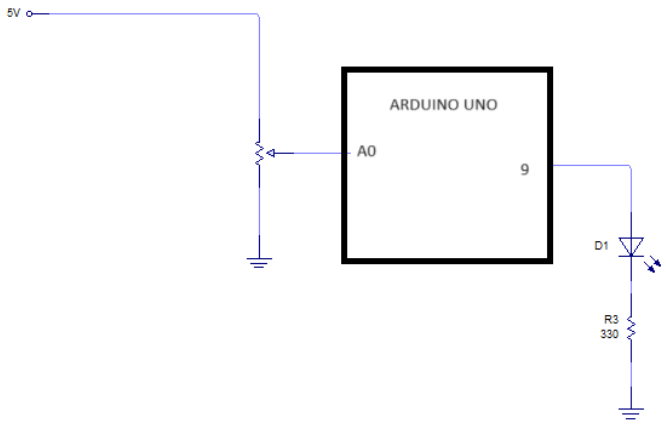


Figura 3.10: Diagrama para el ejemplo de salida PWM.

3.4.4 Conversión digital a analógico

La conversión de digital a analógico (DAC) transforma señales digitales en señales analógicas continuas. Métodos comunes:

- **Conversión Binaria Ponderada:** cada bit tiene un peso asignado según su posición.
- **Conversión R-2R Ladder:** usa resistencias con valores R y 2R para dividir el voltaje de referencia.

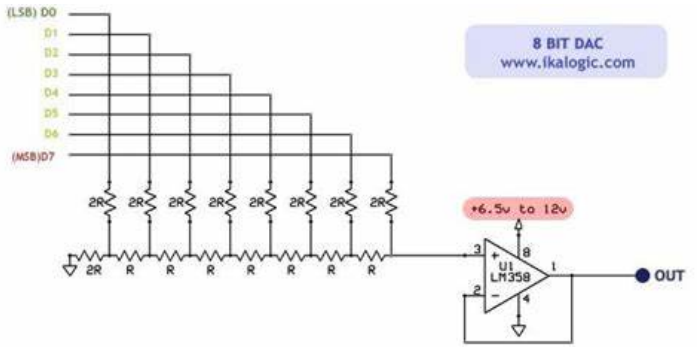


Figura 3.11: Convertidor R-2R.

```
void setup() {
  DDRD = 0xFF; // Configura PORTD como salidas
}

void loop() {
  for (int i = 0; i < 256; i++) {
    PORTD = i;
    delay(10);
  }
}
```

Listing 3.8: Función diente de sierra con DAC R-2R de 8 bits.

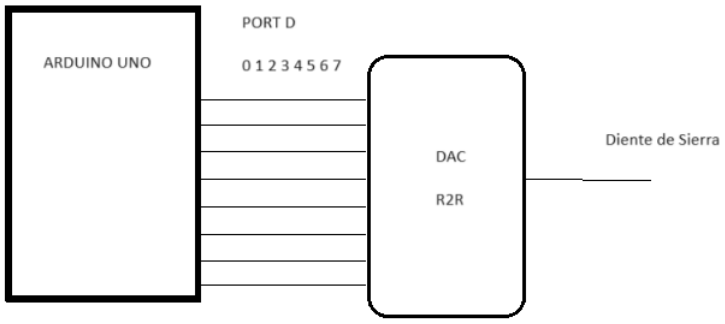


Figura 3.12: Conexión de DAC con Arduino.

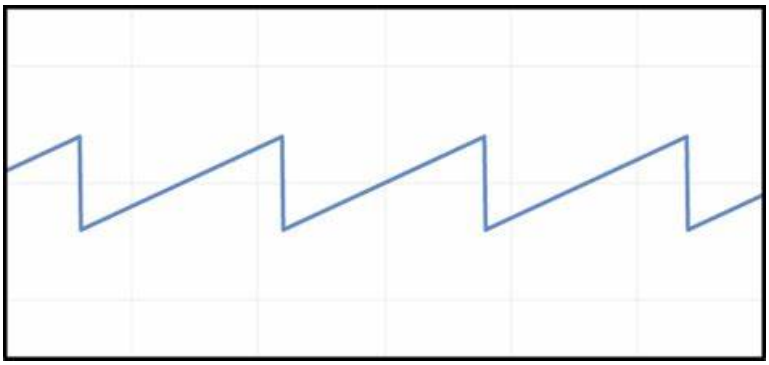


Figura 3.13: Forma de onda esperada en la salida del ejemplo DAC.

3.4.5 Comunicaciones en microcontroladores

Los microcontroladores modernos soportan una amplia variedad de tipos de comunicación:

UART/USART Comunicación serial asíncrona; usado con GPS, GSM, sensores.

SPI Bus síncrono maestro/esclavo de alta velocidad; pantallas LCD, tarjetas SD.

I2C Bus serial para periféricos de baja velocidad; sensores, EEPROMs.

USB Alta velocidad; comunicación con computadoras.

Bluetooth Inalámbrico de corto alcance.

Wi-Fi Inalámbrico de alta velocidad; IoT.

ZigBee/Thread Baja potencia; redes de sensores y domótica.

CAN Automotriz; redes de controladores sin host central.

Modbus Automatización industrial; PLCs.

Ejemplo UART con Arduino UNO

```
void setup() {
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
    char receivedChar = Serial.read();
    if (receivedChar == 'A') {
      digitalWrite(13, HIGH);
    } else if (receivedChar == 'B') {
      digitalWrite(13, LOW);
    }
  }
}
```

Listing 3.9: Comunicación Serial UART con Arduino UNO.

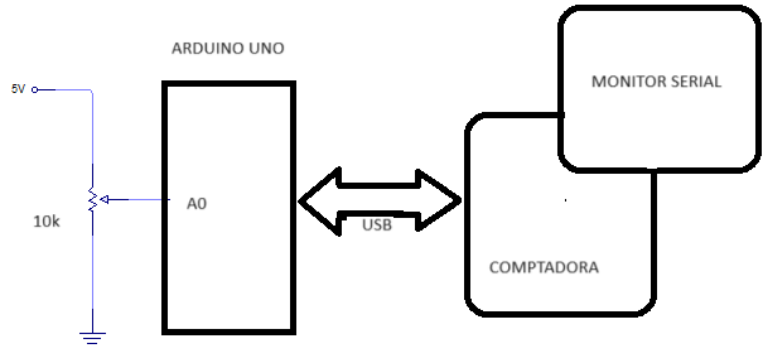


Figura 3.14: Diagrama funcional del ejemplo UART.

Ejemplo I2C con Arduino UNO

```
#include <Wire.h>
#define SLAVE_ADDRESS 0x04
#define ANALOG_PIN A0
#define THRESHOLD_VOLTAGE 3
#define VOLTAGE_RESOLUTION 5.0
#define MAX_ANALOG_READ 1023

void setup() {
  Wire.begin();
  Serial.begin(9600);
}

void loop() {
  int analogValue = analogRead(ANALOG_PIN);
  float voltage = (analogValue * VOLTAGE_RESOLUTION) / MAX_ANALOG_READ
      ;
  Wire.beginTransmission(SLAVE_ADDRESS);
  Wire.write(voltage > THRESHOLD_VOLTAGE ? 1 : 0);
  Wire.endTransmission();
  delay(100);
}
```

Listing 3.10: Maestro I2C — Arduino UNO.

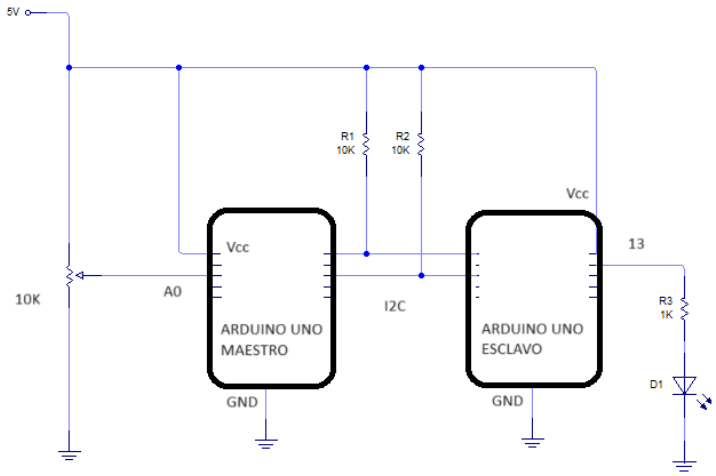


Figura 3.15: Diagrama para el ejemplo de comunicación I2C.

Ejemplo Bluetooth HC-06 con Arduino UNO

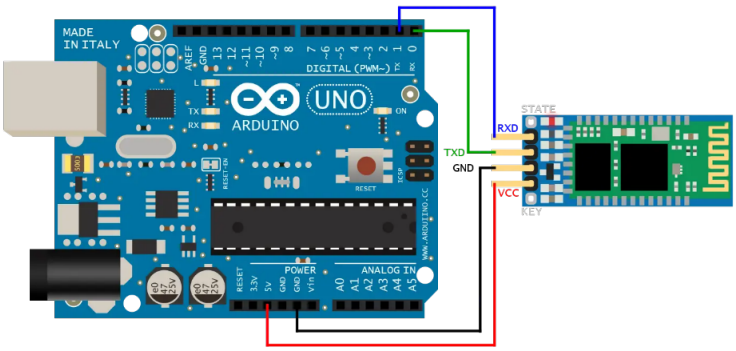


Figura 3.16: Diagrama para el ejemplo de comunicación Bluetooth.

```
int ledPin = 13;

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

void loop() {
  if (Serial.available()) {
    char receivedChar = Serial.read();
    if (receivedChar == '1') {
      digitalWrite(ledPin, HIGH);
    } else if (receivedChar == '0') {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Listing 3.11: Control de LED vía Bluetooth HC-06.

Ejemplo Ethernet Shield con Arduino UNO

Figura 3.17: Arduino UNO conectado a Ethernet Shield.

3.5 Actuadores

Un actuador es un dispositivo que convierte una señal eléctrica, hidráulica o neumática en una acción física o mecánica. Algunos ejemplos comunes:

- **Relevadores:** dispositivos electromecánicos para controlar circuitos de alta potencia con señales de baja potencia.
- **Válvulas solenoides:** controlan el flujo de líquidos o gases.
- **Motores eléctricos:** DC, paso a paso, brushless, servomotores.
- **Cilindros neumáticos e hidráulicos:** movimiento lineal.
- **LEDs y dispositivos de iluminación:** indicadores de estado.

Tarjeta de relevadores

Figura 3.18: Tarjeta de relevadores electromagnéticos (+5 V).

Relevador de Estado Sólido (SSR)

Figura 3.19: Relevador de Estado Sólido (SSR).

Motor DC

Figura 3.20: Motor CC de imán permanente.

Motor Paso a Paso

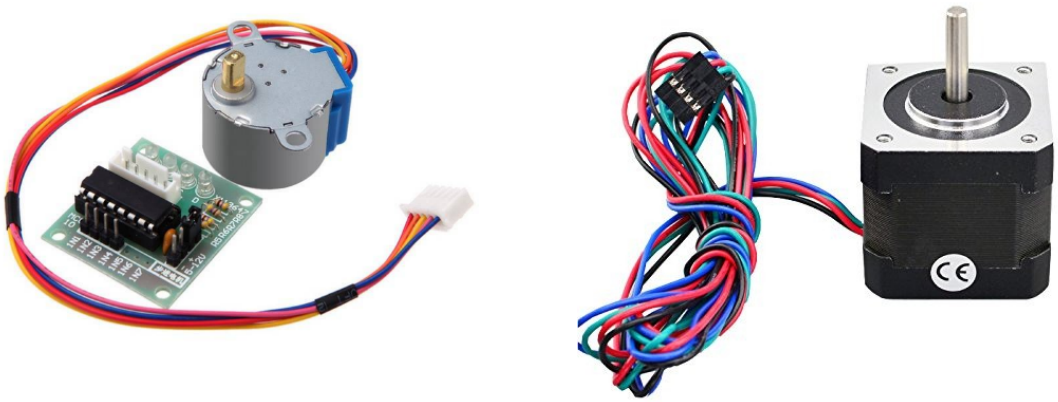


Figura 3.21: Motor paso a paso monopolar y bipolar.

Motor Brushless (BLDC)

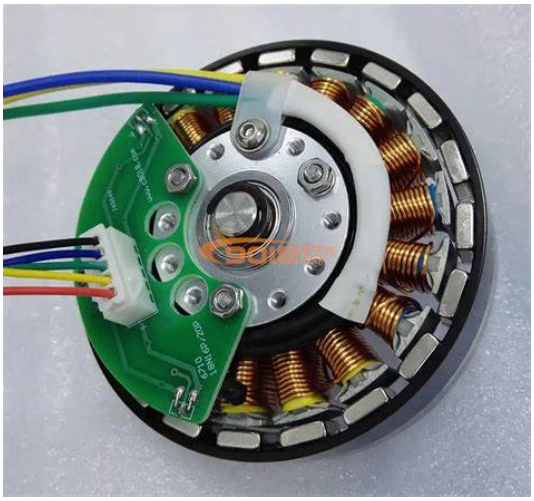


Figura 3.22: Motor Brushless.

Servomotor

Figura 3.23: Servomotor industrial y servomotor para pequeños proyectos.

Motor AC

Figura 3.24: Motor de inducción monofásico.

Ejemplo Arduino con Relevador

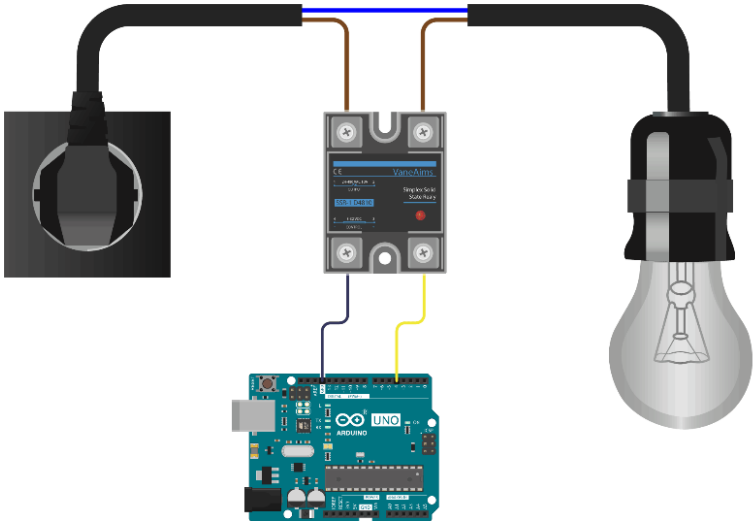


Figura 3.25: Control de foco 120 V AC con Arduino UNO.

Ejemplo Arduino con Servo Motor

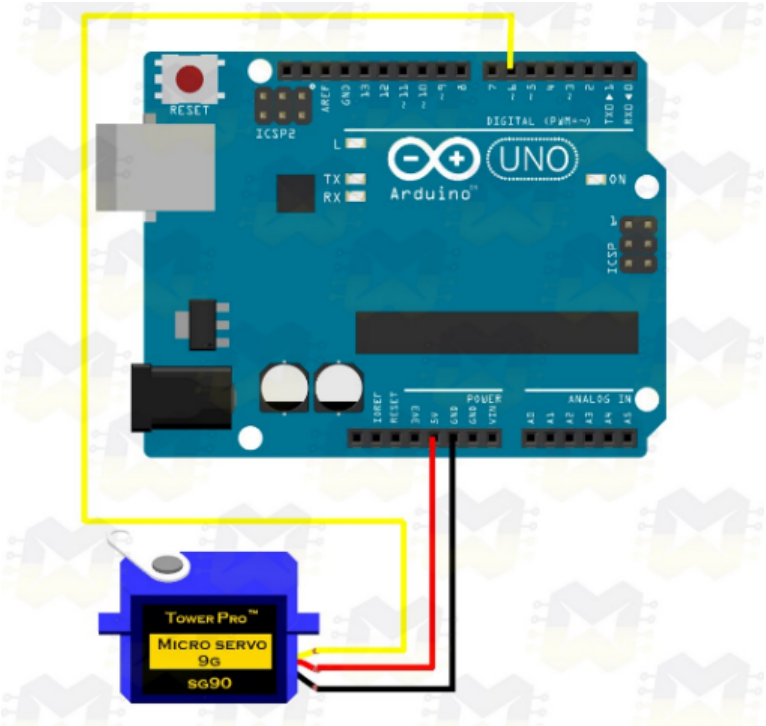


Figura 3.26: Control de servo motor con Arduino.

```
#include <Servo.h>

const int servoPin = 6;
Servo servoMotor;

void setup() {
  servoMotor.attach(servoPin);
}

void loop() {
  servoMotor.write(0);
  delay(1000);
  servoMotor.write(90);
  delay(1000);
  servoMotor.write(180);
  delay(1000);
}
```

Listing 3.12: Control de servomotor con Arduino UNO.

3.6 Sensores

Los sensores son dispositivos que detectan cambios en su entorno y convierten esas mediciones en señales eléctricas que pueden ser interpretadas por un sistema de control. Tipos comunes:

- Temperatura, presión, posición y movimiento

- Luz, fuerza y torque, flujo, nivel
- Humedad y gas

Sensor ultrasónico HC-SR04

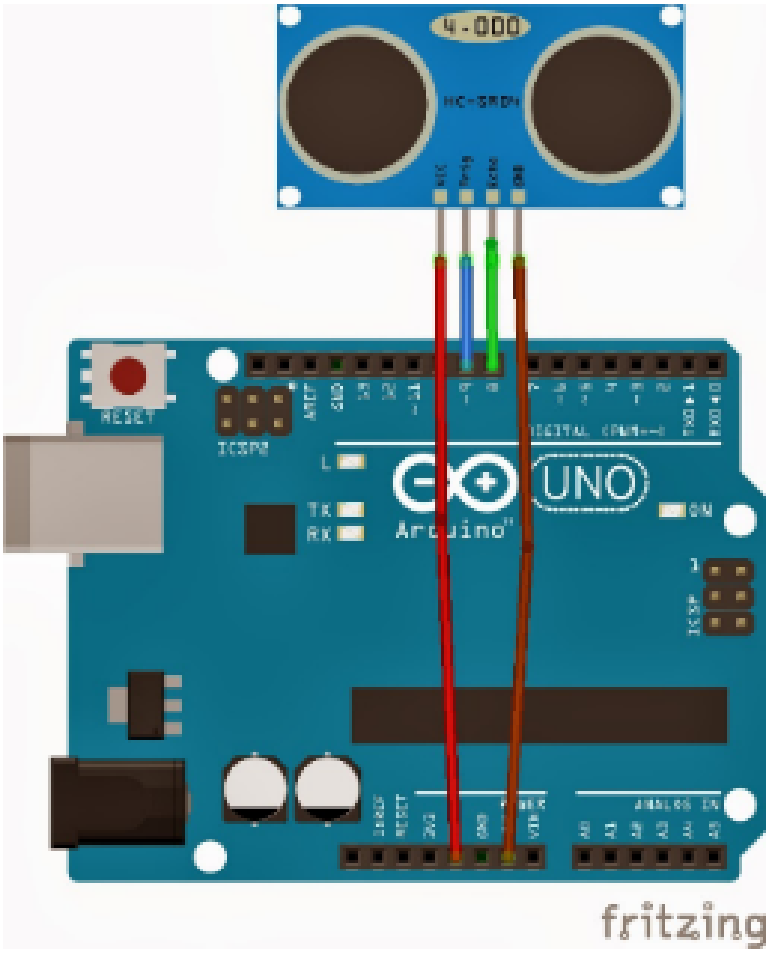


Figura 3.27: Conexión de sensor HC-SR04 con Arduino.

```
long duration;
int distance;
const int trigPin = 9;
const int echoPin = 10;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
```

```
19    distance = duration * 0.034 / 2;
20    Serial.print("Distancia: ");
21    Serial.print(distance);
22    Serial.println(" cm");
23    delay(1000);
24  }
```

Listing 3.13: Arduino con sensor ultrasónico HC-SR04.

Sensor LDR

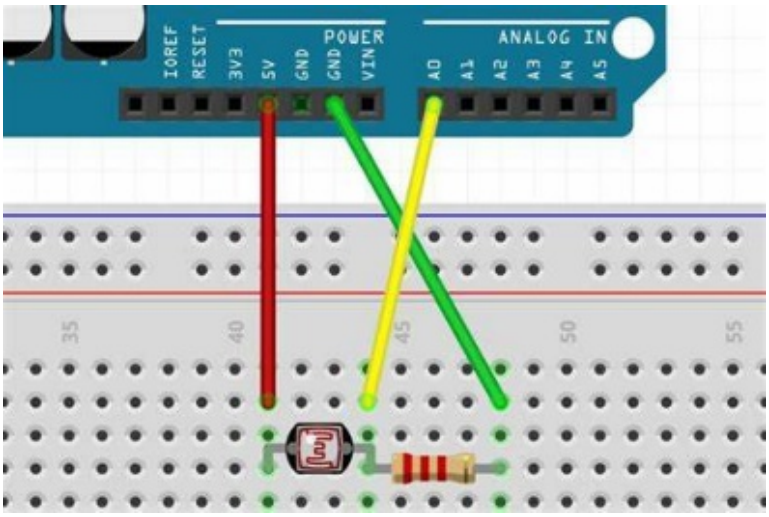


Figura 3.28: Conexión de sensor LDR con Arduino.

3.7 HMI

HMI (*Human-Machine Interface*) o Interfaz Hombre-Máquina se refiere a cualquier dispositivo o software que permite la interacción entre humanos y máquinas en sistemas automatizados.

Ejemplo HMI con Teclado y Display

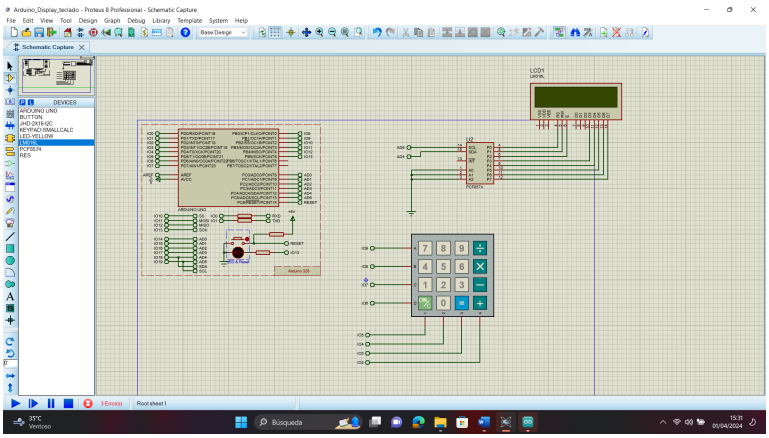


Figura 3.29: Conexión de display y teclado.

```ino
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>

LiquidCrystal_I2C lcd(0x20, 16, 2);

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'\0','0','=','+'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS)
    ;

void setup() {
  lcd.init();
  lcd.backlight();
}

void loop() {
  char key = keypad.getKey();
  if (key) {
    lcd.clear();
    switch (key) {
      case '1': lcd.print("Mensaje 1"); break;
      case '2': lcd.print("Mensaje 2"); break;
      default:  lcd.print("Tecla: "); lcd.print(key); break;
    }
  }
}
```

Listing 3.14: HMI simple con teclado matricial y LCD I2C.

Ejemplo HMI Python (consola)

```python
import serial
import time

arduino = serial.Serial('COM3', 9600)
time.sleep(2)

def toggle_led(command):
    arduino.write(command.encode())

try:
    while True:
        cmd = input("Introduce '1' para encender, '0' para apagar: ")
        if cmd in ['0', '1']:
            toggle_led(cmd)
        else:
            print("Comando no valido.")
except KeyboardInterrupt:
```

```
18      arduino.close()
```

Listing 3.15: HMI Python por consola para controlar un LED vía serial.

Ejemplo HMI Python GUI (Tkinter)

```
1  import tkinter as tk
2  from tkinter import ttk
3  import serial, time
4
5  try:
6      arduino = serial.Serial('COM3', 9600)
7      time.sleep(2)
8  except:
9      print("Verifica la conexion del Arduino.")
10
11 def encender_led():
12     arduino.write(b'1')
13
14 def apagar_led():
15     arduino.write(b'0')
16
17 root = tk.Tk()
18 root.title("Control LED Arduino")
19 root.geometry("300x150")
20
21 btn_encender = ttk.Button(root, text="Encender LED", command=
       encender_led)
22 btn_encender.pack(pady=20)
23 btn_apagar = ttk.Button(root, text="Apagar LED", command=apagar_led)
24 btn_apagar.pack(pady=20)
25
26 root.mainloop()
```

Listing 3.16: HMI Python con GUI usando Tkinter.

HMI Visual Studio (Visual Basic)

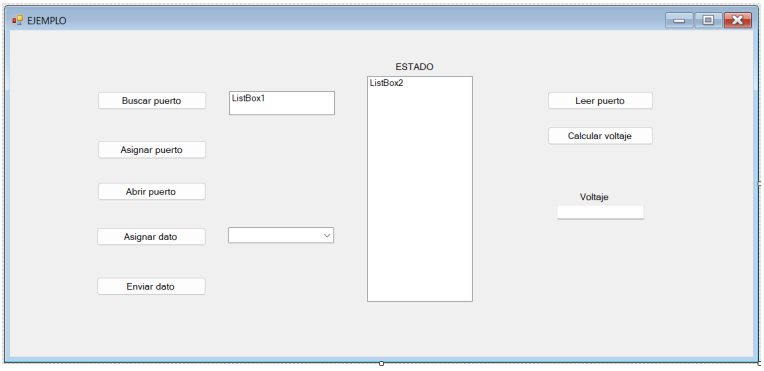


Figura 3.30: Formulario del ejemplo de HMI Visual Studio.

```
1  Public Class Form1
2      Dim dato As Char
```

```
    Dim lectura As String
    Dim voltaje As Single

    Private Sub Button1_Click(...) Handles Button1.Click
        For Each sp As String In My.Computer.Ports.SerialPortNames
            ListBox1.Items.Add(sp)
        Next
    End Sub

    Private Sub Button2_Click(...) Handles Button2.Click
        SerialPort1.PortName = ListBox1.Items.Item(0)
    End Sub

    Private Sub Button5_Click(...) Handles Button5.Click
        SerialPort1.Open()
    End Sub

    Private Sub Button4_Click(...) Handles Button4.Click
        SerialPort1.WriteLine(dato)
    End Sub

    Private Sub Button6_Click(...) Handles Button6.Click
        lectura = SerialPort1.ReadExisting
        ListBox2.Items.Add("Valor leido = " & lectura)
    End Sub

    Private Sub Button7_Click(...) Handles Button7.Click
        voltaje = CSng(lectura) * 5 / 1023
        TextBox1.Text = CStr(voltaje)
    End Sub
End Class
```

Listing 3.17: HMI en Visual Basic .NET para control de LED y lectura analógica.

Capítulo 4

Otros dispositivos programables

Los dispositivos programables abren un vasto mundo de posibilidades para diseñadores, ingenieros y entusiastas de la tecnología. Entre ellos destacan los PLD, los PLC y las Raspberry Pi por su versatilidad y capacidades únicas.

4.1 La computadora Raspberry Pi

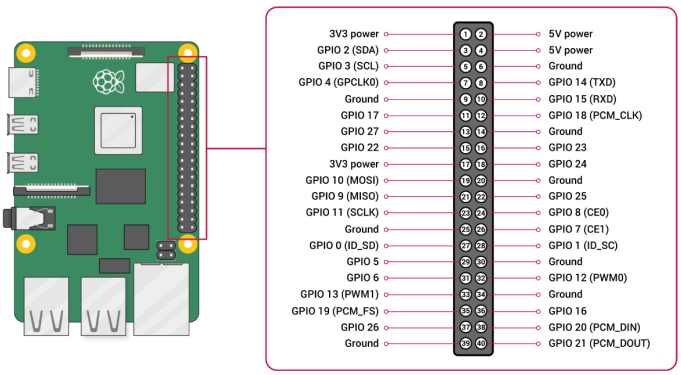


Figura 4.1: Puerto GPIO de Raspberry Pi.

La Raspberry Pi es una serie de pequeñas computadoras de placa única desarrolladas por la Raspberry Pi Foundation en colaboración con Broadcom. Su objetivo es fomentar la enseñanza de ciencias de la computación. A pesar de su pequeño tamaño, puede realizar muchas funciones de una computadora de escritorio: procesamiento de textos, navegación por internet, reproducción de video HD, etc.

Las Raspberry Pi incluyen puertos HDMI, USB, GPIO, ranuras para tarjetas SD, y pueden ejecutar varios sistemas operativos, siendo Raspberry Pi OS el más popular.

4.2 PLD

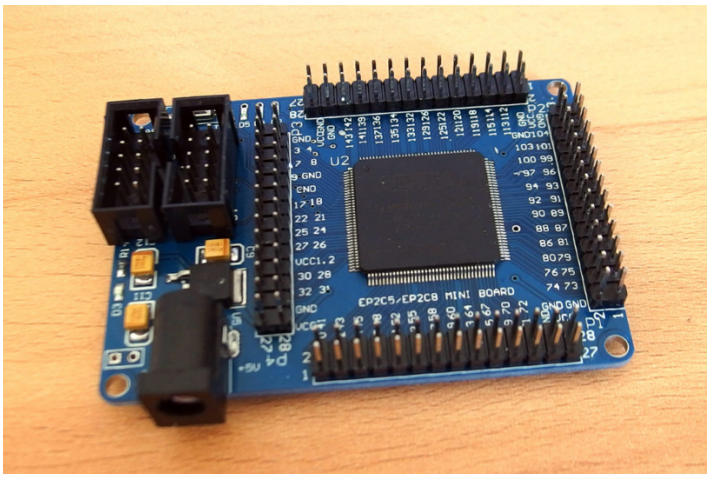


Figura 4.2: Tarjeta entrenador de FPGA.

Un PLD (*Programmable Logic Device*) es un dispositivo electrónico utilizado en el campo del diseño digital que permite configurar su funcionalidad para realizar operaciones lógicas específicas. Los PLDs se programan usando lenguajes de descripción de hardware (HDL) como VHDL o Verilog.

Tipos de PLDs:

PROM / PAL Los más simples; funciones lógicas limitadas.

GAL Similar al PAL, pero reprogramable.

CPLD Mayor complejidad; múltiples bloques lógicos en un chip.

FPGA El mayor nivel de complejidad y flexibilidad; miles o millones de puertas lógicas programables.

Nota: Este tema se profundiza en un libro dedicado exclusivamente a dispositivos lógicos programables.

4.3 PLC

Figura 4.3: PLC.

Un PLC (*Controlador Lógico Programable*) es un tipo de computadora especializada utilizada en el ámbito industrial para controlar procesos de manufactura y maquinaria. Fue diseñado para reemplazar los sistemas de control electromecánicos.

Los lenguajes de programación de PLC según la norma IEC 61131-3 incluyen:

Ladder Diagram (LD) Lenguaje gráfico que emula esquemas eléctricos de circuitos de relés. El más popular.

Function Block Diagram (FBD) Bloques gráficos para representar funciones lógicas.

Structured Text (ST) Similar a los lenguajes de alto nivel.

Sequential Function Chart (SFC) Para programar secuencias de operaciones.

Instruction List (IL) Lista de instrucciones; similar al ensamblador.

Nota: Este tema se profundiza en un libro dedicado exclusivamente a controladores lógicos programables.

Bibliografía

- Tocci, R. J., Widmer, N. S., & Moss, G. L. (2007). *Sistemas digitales: Principios y aplicaciones* (10.ª ed.). Pearson Educación.
- Floyd, T. L. (2008). *Dispositivos electrónicos* (8.ª ed.). Pearson Educación.
- Microchip Technology. (2024). *PIC16F872 Datasheet*. https://www.microchip.com
- Atmel Corporation. (2015). *ATmega328P Datasheet*. https://www.microchip.com
- Arduino. (2024). *Arduino Reference*. https://www.arduino.cc/reference/en/
- Espressif Systems. (2024). *ESP32 Technical Reference Manual*. https://www.espressif.com
- Raspberry Pi Foundation. (2024). *Raspberry Pi Pico Datasheet*. https://www.raspberrypi.com

Créditos de figuras

Fig.	Descripción	Fuente
1	Buffer	Elaboración propia
2–9	Compuertas lógicas	Tocci et al. (2007)
10–14	Ejemplos álgebra booleana	Tocci et al. (2007)
15–19	Mapas de Karnaugh	Tocci et al. (2007)
20–25	Circuitos MSI	Tocci et al. (2007)
26–33	Circuitos aritméticos	Tocci et al. (2007)
34–37	Flip-Flops	Tocci et al. (2007)
38	Contadores 74ALS16X	Tocci et al. (2007)
39	Contador BCD	https://mikitronic.blogspot.com
40	Registro 74LS374	http://www.usbekits.com
41	EEPROM 28C256	https://theansweris27.com
42	Arquitectura Von Neumann	https://www.proprofs.com
43	Arquitectura Harvard	https://www.trccompsci.online
44	ALU 74HC382	Tocci et al. (2007)
45	Interconexión de Buses	https://arquitecturadecomputadorasrolon.blogspot.com
46	Zilog Z80	https://www.tradera.com
47	Intel 8086	https://shilpadotcom.blogspot.com
48	Intel 8088	https://www.grupelektronik.com
49	Motorola 68000	https://hipertextual.com
50	Zócalo LGA2066	https://www.partitionwizard.com
51	Zócalo sTRX4	https://www.chaynikam.info
52	Microprocesador ARM	https://iot.electronicsforu.com
53	Tarjeta Madre	https://quizlet.com
54	PIC16F872	https://www.futurlec.com
55	ATMEGA328P	https://www.sigmaelectronica.net
56	Arduino UNO	https://mamaseh.blogspot.com
57	Raspberry Pi Pico	https://uelectronics.com
58	ESP32	https://programarfacil.com
59–61, 63, 65–68	Diagramas de ejemplos	Elaboración propia

Fig.	Descripción	Fuente
62	PWM ciclos de trabajo	http://www.healyourselfathome.com
64	Convertidor R2R	https://embeddednewbie.blogspot.com
69	Bluetooth HC-06	https://linhkienhatech.com
70	Ethernet Shield	https://programarfacil.com
71	Tarjeta relevadores	https://pigra.com.mx
72	SSR	Amazon
73	Motor DC	https://toboids.com
74	Motor paso a paso	http://arduino.vn, MercadoLibre
75	Brushless Motor	https://www.pinterest.pt
76	Servomotores	https://www.tecservautomacao.com, https://maxelectronica.cl
77	Motor AC	https://www.tradeindia.com
78	Control foco AC	https://chistotnik.ru
79	Control servo	https://blogmasterwalkershop.com.br
80	HC-SR04	https://www.electrontools.com
81	LDR	https://www.arrow.com
82	Display y teclado	https://marduino.ma, https://inputmakers.com
83	HMI Visual Studio	Elaboración propia
84	GPIO Raspberry Pi	https://enfow.github.io
85	Entrenador FPGA	https://educ8s.tv
86	PLC	https://www.machinemetrics.com