



Diseño y Desarrollo de Soluciones IOT

**MATERIAL TÉCNICO
DE APOYO**

TAREA N°01

Revisa los fundamentos de IoT y plataformas de hardware.

1. FUNDAMENTOS DE ELECTRÓNICA DIGITAL.

- **Introducción.**

La electrónica digital es la rama de la electrónica más moderna y que evoluciona más rápidamente. Se encarga de sistemas electrónicos en los que la información está codificada en estados discretos, a diferencia de los sistemas analógicos donde la información toma un rango continuo de valores.

Estos son algunos conceptos clave en electrónica digital:

- **Bit:** Es la unidad básica de información en electrónica digital y puede tener dos valores: 0 o 1.
- **Lógica Booleana:** Utiliza operadores lógicos como AND, OR y NOT para realizar operaciones con valores binarios.
- **Puertas lógicas:** Son los componentes fundamentales en los circuitos digitales que implementan operaciones lógicas (por ejemplo, AND, OR, XOR).
- **Flip-Flop:** Un circuito que puede almacenar un bit de información, usado en registros y contadores.
- **Microprocesadores y microcontroladores:** Dispositivos digitales que procesan datos y ejecutan instrucciones en forma de código binario.



[Electrónica digital](#)

- **Voltaje**

El voltaje se puede definir como la magnitud responsable de crear la diferencia de potencial eléctrico entre dos puntos. Por esta razón, a menudo se le denomina tensión eléctrica o diferencia de potencial eléctrico.

En un contexto más técnico, el voltaje se refiere al trabajo realizado por una unidad de carga eléctrica cuando se desplaza a través de dos puntos en un campo eléctrico específico.

Para entender mejor este concepto hay que repasar ciertas palabras clave y sus significados. La electricidad se basa en tres conceptos prácticos, los vatios, los amperios y los voltios.

- **Voltaje inducido.**

Este tipo de voltaje se genera para alimentar un circuito eléctrico específico. Cuando el circuito está abierto, este voltaje puede mantener la diferencia de potencial entre dos puntos. Sin embargo, si el circuito está cerrado, se producirá un flujo de corriente eléctrica.

- **Voltaje alterno.**

El voltaje alterno, abreviado como VA, se representa en gráficos cartesianos con un eje negativo y uno positivo, lo que resulta en una forma de onda sinusoidal. Este tipo de voltaje es comúnmente utilizado en las tomas de corriente, ya que es fácil de transportar y generar. Lo característico del VA son sus valores alternantes, lo que da origen a su nombre. La constancia en el tiempo de este voltaje puede variar según el país o región en el que se encuentre.

- **Voltaje de corriente continua.**

Este tipo de voltaje se encuentra con mayor frecuencia en baterías y motores, ya que se puede obtener al transformar una corriente alterna en una corriente continua. Sin embargo, no es necesario que sea perfectamente constante, ya que, mediante transformadores y fusibles, puede presentar algunas fluctuaciones que no son lo suficientemente grandes como para considerarlo un voltaje alterno.

- **Voltaje continuo.**

El voltaje continuo, conocido como VCC (voltaje de corriente continua), es aquel en el que la corriente es constante y no presenta fluctuaciones. Se encuentra comúnmente en microprocesadores o chips, ya que estos dispositivos requieren un voltaje constante para funcionar de manera adecuada. Para obtener este tipo de voltaje, generalmente se requiere un tratamiento utilizando condensadores electrolíticos.

- **Resistencia**

Es un componente electrónico pasivo que se utiliza para limitar o regular la corriente eléctrica en un circuito. Su función principal es reducir la cantidad

de corriente que fluye a través de un circuito, disipando energía en forma de calor. Esto se logra porque la resistencia opone el paso de la corriente eléctrica, de acuerdo con la Ley de Ohm, que establece la relación entre el voltaje (V), la corriente (I) y la resistencia (R) de la siguiente forma:

$$V = I \times R$$

Donde:

V: es el voltaje (en voltios),

I: es la corriente (en amperios),

R: es la resistencia (en ohmios).

- **Características principales de una resistencia:**

- **Valor de resistencia:** Se mide en ohmios (Ω) y determina cuánta oposición ofrece al paso de la corriente.
- **Tolerancia:** Indica el margen de error en el valor de la resistencia, expresado como un porcentaje.
- **Potencia:** Se refiere a la cantidad de energía que la resistencia puede disipar sin dañarse, medida en vatios (W).

- **Tipos comunes de resistencias:**

- **Fijas:** Tienen un valor de resistencia constante.
- **Variables (potenciómetros):** Permiten ajustar su valor de resistencia manualmente.
- **Resistencias dependientes de temperatura (NTC/PTC):** Su valor varía con la temperatura.

- **Amperaje**

El amperaje (también llamado corriente eléctrica) es la cantidad de carga eléctrica que fluye a través de un conductor por unidad de tiempo. Se mide en amperios (A), que es la unidad del Sistema Internacional para la corriente eléctrica.

Un amperio representa el flujo de un coulomb de carga eléctrica por segundo. Es decir, si un amperio de corriente está fluyendo, significa que un coulomb de carga (aproximadamente 6.24×10^{18} electrones) pasa por un punto del conductor en un segundo.

- **Fórmula para el amperaje:**

La corriente eléctrica se puede calcular utilizando la Ley de Ohm cuando se conocen el voltaje y la resistencia:

$$I = V / R$$

Donde:

I: es la corriente (en amperios).

V: es el voltaje (en voltios).

R: es la resistencia (en ohmios).

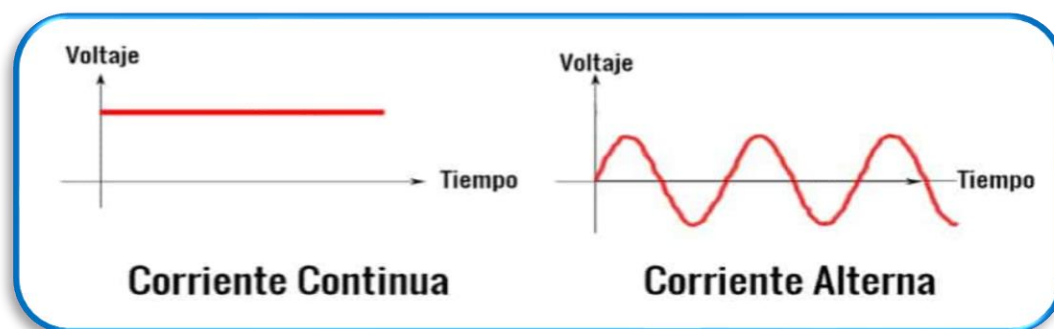
- **Tipos de corriente:**

- **Corriente continua (DC):** El flujo de electrones es en una sola dirección.

- **Corriente alterna (AC):** El flujo de electrones cambia de dirección periódicamente.

- **Importancia del amperaje:**

El amperaje es crucial para dimensionar componentes en un circuito eléctrico o electrónico. Si un circuito lleva más corriente de la que un componente puede manejar, este puede sobrecalentarse y dañarse. Además, en sistemas eléctricos, como el cableado de una casa, el amperaje determina el grosor necesario de los cables para evitar peligros.



[Tipos de corriente](#)

2. FUNDAMENTOS DE HARDWARE LIBRE.

El hardware libre (o hardware de código abierto) se refiere a dispositivos físicos cuyas especificaciones de diseño son públicas, lo que permite a cualquier persona estudiarlos, modificarlos, distribuirlos y producirlos. Los planos, esquemas y documentación técnica son accesibles de forma gratuita o con licencias que garantizan su uso y modificación, promoviendo la colaboración y el aprendizaje.

- **Características del hardware libre:**

- **Diseño abierto:** Toda la información necesaria para reproducir el hardware está disponible para el público.

- **Licencia abierta:** Los esquemas, planos y firmware suelen estar licenciados bajo términos que permiten su libre uso, modificación y distribución.
- **Colaboración:** Permite a la comunidad mejorar los diseños originales y adaptarlos a nuevas necesidades.
- **Transparencia:** Se puede verificar cómo funciona el hardware, lo que es útil en términos de seguridad y ética tecnológica.
- **Tipos de hardware libre:**
 - **Placas de desarrollo:** Dispositivos usados por desarrolladores y entusiastas para crear prototipos y proyectos electrónicos.
 - **Impresoras 3D:** Diseños abiertos que permiten a los usuarios crear sus propias impresoras 3D y modificarlas.
 - **Sistemas embebidos:** Hardware diseñado para tareas específicas, pero con esquemas y planos abiertos.
 - **Computadoras de placa única (SBC):** Dispositivos completos en una sola placa, con un microprocesador, memoria y almacenamiento integrados.
 - **Periféricos:** Hardware adicional para dispositivos informáticos como teclados, ratones, pantallas, etc., con especificaciones abiertas.
- **Ejemplos de hardware libre:**
 - **Arduino:** Uno de los ejemplos más populares. Es una plataforma de hardware libre basada en microcontroladores, ideal para proyectos de electrónica y robótica.
 - **Raspberry Pi:** Aunque no es completamente libre (el diseño de su procesador es propietario), gran parte de su software y hardware es abierto, y es muy utilizada en la comunidad de hardware libre para proyectos educativos y de prototipado.
 - **Prusa i3:** Una impresora 3D de código abierto que ha sido ampliamente replicada y mejorada por la comunidad.
 - **BeagleBone:** Es una computadora de placa única (SBC) diseñada para aplicaciones de desarrollo y prototipos, cuyos esquemas están disponibles al público.
 - **Open Source Hardware Association (OSHWA):** Una organización que certifica proyectos de hardware libre. Certifican que proyectos como la

impresora 3D LulzBot o el reloj inteligente Bangle.js cumplen con los estándares de hardware abierto.

- **Ventajas del hardware libre:**

- **Autonomía:** Permite que los usuarios entiendan y modifiquen completamente el hardware.
- **Accesibilidad:** Facilita la creación de hardware a bajo costo.
- **Innovación rápida:** Gracias a la colaboración abierta, el desarrollo y la mejora son más rápidos.

El hardware libre fomenta una cultura de compartir conocimiento y empodera a las personas para diseñar, crear y modificar tecnología según sus necesidades.

3. FUNDAMENTOS DE COMUNICACIÓN SERIAL Y PARALELA.

La comunicación serial es un método de transferencia de datos entre dispositivos electrónicos, donde la información se envía bit a bit, en secuencia, a través de un solo canal o cable de comunicación. A diferencia de la comunicación paralela, que transmite varios bits simultáneamente a través de múltiples canales, la comunicación serial utiliza menos cables, lo que la hace más simple y económica, especialmente para largas distancias.

Existen dos tipos principales de comunicación serial:

- **Asíncrona:** Los datos se envían en cualquier momento y se usan bits de inicio y parada para indicar el comienzo y el fin de cada paquete de datos. Un ejemplo común es la comunicación a través de puertos RS-232 o UART.
- **Síncrona:** Los datos se envían de manera continua junto con una señal de reloj que sincroniza al emisor y al receptor. Ejemplos incluyen I2C y SPI.

Este tipo de comunicación se utiliza en aplicaciones como la transmisión de datos entre microcontroladores, computadoras y periféricos, como sensores o módulos de comunicación (Wi-Fi, Bluetooth).

La comunicación paralela es un método de transferencia de datos en el que varios bits se envían simultáneamente a través de múltiples líneas de datos (canales o cables) de manera coordinada.

Cada línea transporta un bit de información, lo que permite transmitir múltiples bits en un solo ciclo de reloj, aumentando la velocidad de transmisión en comparación con la comunicación serial, que envía los bits uno tras otro por un solo canal.

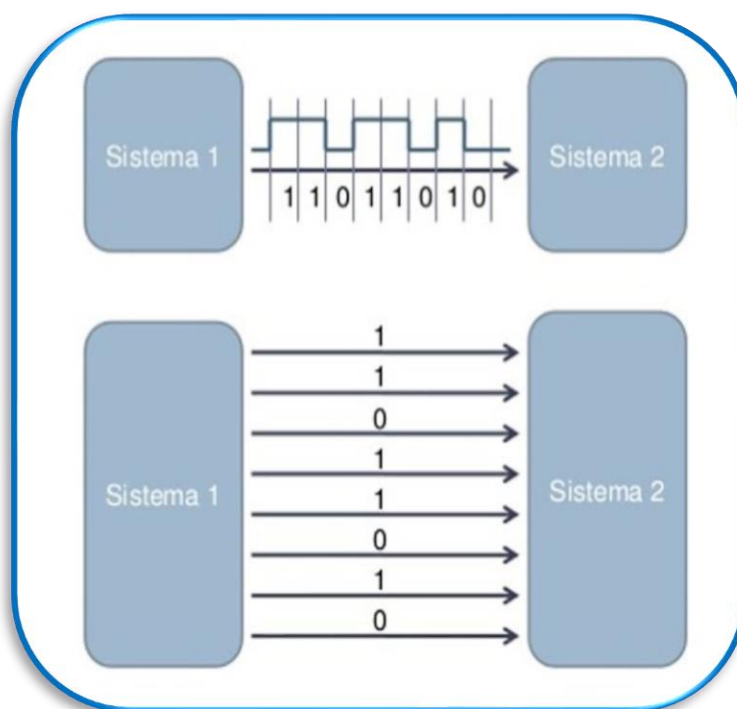
- **Características de la comunicación paralela:**

- **Velocidad:** Es más rápida que la comunicación serial para distancias cortas, ya que varios bits se transmiten simultáneamente.
- **Número de cables:** Se requieren más cables o líneas físicas, ya que cada bit tiene su propio canal.
- **Sincronización:** Al igual que la comunicación serial síncrona, suele utilizar una señal de reloj que coordina el envío y la recepción de datos.

- **Ejemplos de comunicación paralela:**

- **Buses de datos en computadoras:** Como el bus de direcciones y datos de un procesador.
- **Puerto paralelo:** Utilizado en impresoras antiguas, como el puerto Centronics.
- **Memorias de computadora:** En la comunicación entre el procesador y la memoria RAM.

Aunque la comunicación paralela puede ser rápida, a largas distancias puede presentar problemas de sincronización y costos más elevados debido al número de cables necesarios, por lo que ha sido sustituida en muchas aplicaciones por la comunicación serial de alta velocidad (como USB o SATA).



Comunicación serial y paralelo

4. ELEMENTOS DE ELECTRÓNICA BÁSICA.

En la electrónica básica, hay varios componentes esenciales que se utilizan para construir circuitos electrónicos. A continuación, te describo los elementos más comunes:

- **Resistencia fija y variable.**

La resistencia es un componente electrónico pasivo cuya función principal es oponer o limitar el flujo de corriente eléctrica en un circuito. La resistencia convierte parte de la energía eléctrica en calor, lo que ayuda a controlar la cantidad de corriente que pasa a través de otros componentes.

- **Características de la resistencia:**

- **Unidad de medida:** Se mide en ohmios (Ω). Un ohmio es la resistencia que permite que una corriente de 1 amperio pase por un conductor cuando se aplica una tensión de 1 voltio.
- **Ley de Ohm (Ω):** La resistencia está relacionada con la corriente (I) y el voltaje (V) mediante la fórmula:

$$V = I * R$$

Donde:

V: es el voltaje en voltios.

I: es la corriente en amperios.

R: es la resistencia en ohmios.

Esta ley muestra que, a mayor resistencia, menor será la corriente para un mismo voltaje.

- **Tipos de resistencias:**

- **Resistencias fijas:** Tienen un valor de resistencia constante.
- **Resistencias variables o potenciómetros:** Permiten ajustar el valor de resistencia de manera manual.
- **Resistencias dependientes de la temperatura (NTC, PTC):** Su valor cambia según la temperatura.
- **Símbolo en los diagramas:** En los esquemas eléctricos, se representa generalmente como una línea zigzagueante (en EE.UU.) o un rectángulo (en Europa).



[Simbología](#)

○ **Aplicaciones de la resistencia:**

- **Limitación de corriente:** Controlan el flujo de corriente en circuitos para proteger otros componentes, como LEDs o transistores.
- **División de voltaje:** En circuitos de divisores de tensión, distribuyen el voltaje de acuerdo con sus valores.
- **Disipación de energía:** Al convertir la energía eléctrica en calor, las resistencias disipan energía, lo que es útil en aplicaciones de carga o regulación.
- El código de colores en las resistencias es un sistema que se utiliza para identificar el valor de su resistencia, su tolerancia y, en algunos casos, su coeficiente de temperatura. Este código está representado por una serie de bandas de colores pintadas en el cuerpo de la resistencia.

✓ **Bandas de colores:**

Las resistencias generalmente tienen 4, 5 o 6 bandas de colores, y cada color tiene un valor numérico asignado.

MECATRONIUM CHIPS		CÓDIGO DE COLORES DE RESISTENCIAS			
COLOR	VALOR 1	VALOR 2	VALOR 3	MULTIPLICADOR	TOLERANCIA
NEGRO	0	0	0	1Ω	± 1%
MARRÓN	1	1	1	10Ω	± 2%
ROJO	2	2	2	100Ω	
NARANJA	3	3	3	1KΩ	
AMARILLO	4	4	4	10KΩ	
VERDE	5	5	5	100KΩ	± 0.5%
AZUL	6	6	6	1MΩ	± 0.25%
VIOLETA	7	7	7	10MΩ	± 0.10%
GRIS	8	8	8		± 0.05%
BLANCO	9	9	9		
ORO				0.1Ω	± 5%
PLATA				0.01Ω	± 10%

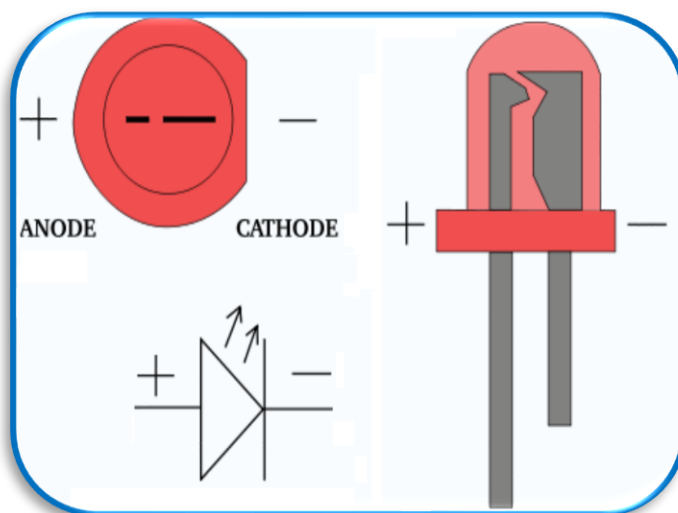
[Código de colores](#)

○ **Leds.**

Un LED (diodo emisor de luz, por sus siglas en inglés Light Emitting Diode) es un tipo especial de diodo que emite luz cuando pasa corriente eléctrica a través de él. A diferencia de los diodos convencionales, que simplemente permiten que la corriente fluya en una sola dirección, el LED convierte parte de la energía eléctrica en luz visible (o infrarroja o ultravioleta, dependiendo del material y la construcción).

- **Características principales de un LED:**

- **Emisión de luz:** Los LEDs emiten luz cuando los electrones se combinan con huecos dentro del material semiconductor, liberando energía en forma de fotones (luz). El color de la luz depende del material semiconductor utilizado.
- **Polaridad:** Al igual que los diodos, los LEDs son dispositivos de polaridad, lo que significa que solo permiten que la corriente fluya en una dirección (del ánodo al cátodo). Si se conecta al revés, el LED no emitirá luz.
- **Bajo consumo de energía:** Comparados con otras fuentes de luz, como las bombillas incandescentes, los LEDs consumen mucha menos energía, lo que los hace más eficientes.
- **Larga vida útil:** Los LEDs pueden durar mucho más tiempo que otras fuentes de luz, ya que no tienen filamentos que se desgasten.
- **Variedad de colores:** Dependiendo del material semiconductor y las técnicas de fabricación, los LEDs pueden emitir diferentes colores, como rojo, verde, azul, amarillo, blanco e incluso colores infrarrojos y ultravioletas.
- **Durabilidad:** Son componentes muy resistentes y pueden soportar golpes, vibraciones y condiciones climáticas extremas.

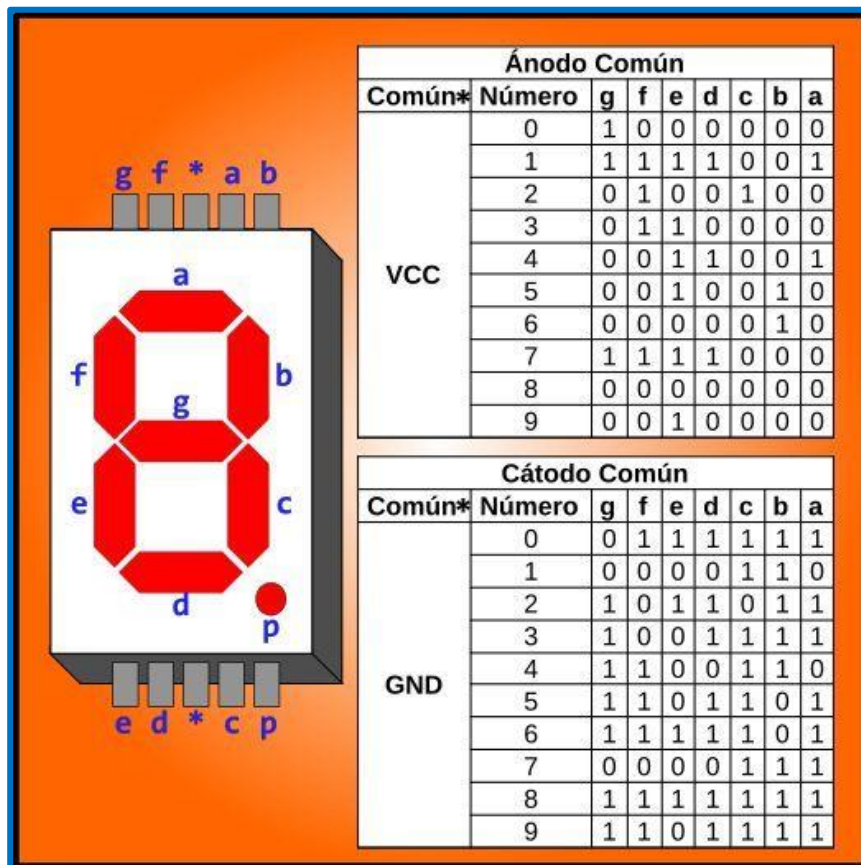


[Diodo led](#)

- **Display de 7 segmentos.**

El display 7 Segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9. Existen dos tipos de display, de cátodo común y de ánodo común. Este tipo de elemento de salida digital o display, se utilizaba en los primeros dispositivos electrónicos de la década de los 70's y 80's.

Es importante mencionar que los display de 7 segmentos, dado que están contruidos con diodos LED, requieren una corriente máxima. En otras palabras, se requiere colocar una resistencia para limitar la corriente. Dicha resistencia depende de la corriente que se quiera suministrar al LED, así como de la caída de voltaje.



[Display de 7 segmentos](#)

○ **Protoboard**

Un protoboard (también conocido como placa de pruebas o breadboard) es una herramienta que se utiliza para construir y probar circuitos electrónicos de manera temporal, sin necesidad de soldar los componentes. Es ideal para proyectos de prototipos, enseñanza y experimentación en electrónica, ya que permite conectar y desconectar componentes de forma rápida y sencilla.

○ **Estructura del protoboard:**

Un protoboard está compuesto por una base de plástico con filas y columnas de agujeros conectados internamente por conductores metálicos. Está dividido en dos partes principales:

○ **Regletas de alimentación (buses):**

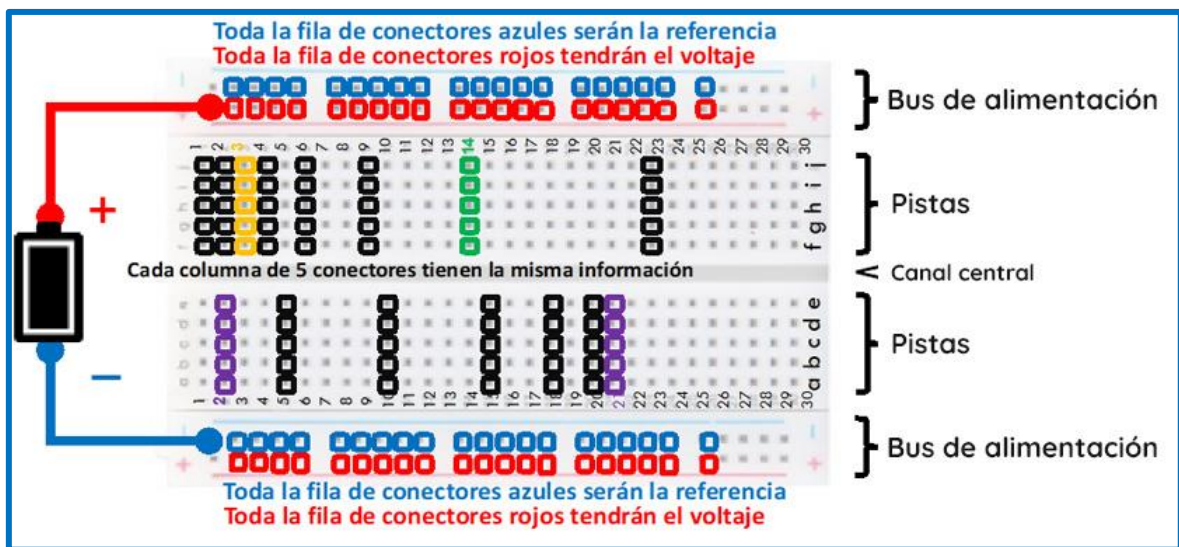
Estas se encuentran a lo largo de los bordes superior e inferior de la placa. Se utilizan para conectar la fuente de energía (voltaje y tierra). Están organizadas en dos columnas, una para el voltaje positivo y otra para el negativo o tierra (GND).

- **Áreas de trabajo:**

La parte central está dividida en varias filas y columnas de agujeros donde se insertan los componentes electrónicos (resistencias, diodos, transistores, LEDs, etc.). Los agujeros de cada fila están conectados eléctricamente en grupos de cinco agujeros, formando un punto de conexión común.

- **Conexiones internas:**

Las filas horizontales en las áreas de trabajo están conectadas entre sí en grupos de cinco orificios. Las columnas verticales en las regletas de alimentación están conectadas entre sí a lo largo de toda la columna (para distribuir la alimentación).



Protoboard

TAREA N°02

Programa hardware con Arduino.

1. ALGORITMOS Y PROGRAMACIÓN CON ARDUINO.

Un algoritmo en programación es un conjunto de instrucciones o pasos bien definidos y secuenciales que se siguen para resolver un problema específico o realizar una tarea determinada. En esencia, un algoritmo es la base de cualquier programa de computadora, ya que define cómo deben procesarse los datos para obtener el resultado deseado.

Un buen algoritmo debe de poseer las siguientes características:

- **Definido:** Cada paso debe estar claramente especificado.
- **Finito:** Debe tener un número limitado de pasos y terminar en un tiempo razonable.
- **Preciso:** Las instrucciones deben ser claras y sin ambigüedades.
- **Eficiente:** Debe optimizar el uso de recursos (tiempo, memoria, etc.).
- **Entradas y salidas:** Debe tener una o más entradas, y generar una o más salidas.

Además, dentro de los algoritmos, se imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar. Es por ello, que se utilizan:

- **Tipos de datos:**
 - **Enteros:**
Se utilizan para almacenar números enteros, positivos o negativos.
 - **int:**
 - ✓ **Tamaño:** 2 bytes (16 bits).
 - ✓ **Rango:** -32,768 a 32,767.
 - ✓ **Ejemplo:** int x = 500;
 - **unsigned int:**
 - ✓ **Tamaño:** 2 bytes (16 bits).
 - ✓ **Rango:** 0 a 65,535 (solo positivos).
 - ✓ **Ejemplo:** unsigned int y = 1000;

- **long:**
 - ✓ **Tamaño:** 4 bytes (32 bits).
 - ✓ **Rango:** -2,147,483,648 a 2,147,483,647.
 - ✓ **Ejemplo:** long z = 100000;
- **unsigned long:**
 - ✓ **Tamaño:** 4 bytes (32 bits).
 - ✓ **Rango:** 0 a 4,294,967,295.
 - ✓ **Ejemplo:** unsigned long t = 3000000000;
- **short:**
 - ✓ **Tamaño:** 2 bytes (16 bits).
 - ✓ **Rango:** -32,768 a 32,767.
 - ✓ **Ejemplo:** short a = -100;
- **unsigned short:**
 - ✓ **Tamaño:** 2 bytes (16 bits).
 - ✓ **Rango:** 0 a 65,535.
 - ✓ **Ejemplo:** unsigned short b = 200;
- **Números de punto flotante:**

Se utilizan para representar números reales o decimales.

 - **float:**
 - ✓ **Tamaño:** 4 bytes (32 bits).
 - ✓ **Rango:** $\pm 3.4028235E+38$ (6 a 7 dígitos de precisión).
 - ✓ **Ejemplo:** float f = 3.14;
 - **double:**
 - ✓ **Tamaño:** 4 bytes (igual que float en Arduino).
 - ✓ En plataformas como el Arduino Uno, float y double tienen el mismo tamaño.
 - ✓ **Ejemplo:** double d = 3.14159;
- **Caracteres:**

Se utilizan para representar un solo carácter o una cadena de caracteres.

 - **char:**
 - ✓ **Tamaño:** 1 byte (8 bits).
 - ✓ **Rango:** -128 a 127 (incluye caracteres ASCII).
 - ✓ **Ejemplo:** char c = 'A';
 - **unsigned char:**
 - ✓ **Tamaño:** 1 byte (8 bits).
 - ✓ **Rango:** 0 a 255.
 - ✓ **Ejemplo:** unsigned char uc = 255;

➤ **String:**

Se utiliza para representar cadenas de texto. Es una clase que facilita la manipulación de secuencias de caracteres.

✓ **Ejemplo:** String greeting = "Hello";

○ **Booleanos:**

Se utilizan para representar valores lógicos, true o false.

➤ **bool:**

✓ **Tamaño:** 1 byte (8 bits, aunque solo usa 1 bit en realidad).

✓ **Valores:** true (1) o false (0).

✓ **Ejemplo:** bool flag = true;

● **Condicionales.**

Las sentencias condicionales son una de las herramientas más útiles en la programación de Arduino. Permiten controlar el flujo de un programa en función de ciertas condiciones que puedes definir en el código. Las sentencias condicionales son como una prueba que comprueba si una condición es verdadera o no. Si la condición es verdadera, el código dentro de la sentencia condicional se ejecuta. Si la condición es falsa, el código no se ejecuta. La sentencia condicional más utilizada es la sentencia if.

○ **Sentencias “if”**

Las sentencias if suelen colocarse en la sección loop(), donde se evalúan una vez en cada ciclo del bucle. El código de una sentencia if tiene el siguiente aspecto:

```
if(sentencia){

}
```

La condición es lo que determina si el código en el cuerpo se ejecuta. Normalmente la condición se escribe para comprobar si una variable es menor, mayor o igual que algún número. Cualquiera que sea la condición utilizada, debe evaluarse a un valor verdadero o falso. Si la condición es verdadera, el programa entrará en las llaves y ejecutará el código dentro del cuerpo. Si la condición es falsa, el programa se salta el código y continúa con la línea de código después de la llave de cierre.

La condición de esta sentencia if es $x < y$. Digamos que x es igual a 2 e y es igual a cuatro. $2 < 4$ es una declaración verdadera, por lo que el programa entrará en las llaves y ejecutará el código en el cuerpo. El código del cuerpo imprimirá en serie “ x es menor que y ” en el monitor serie.

Si x es igual a 10 e y es igual a cuatro, $10 < 4$ es una declaración falsa, por lo que el programa omitirá el código en las llaves y no se imprimirá nada en el monitor serial.

○ Operadores relacionales

Los operadores relacionales son muy útiles para escribir sentencias condicionales. Los operadores relacionales se usan para comparar dos o más números o variables. Por ejemplo, igual a, mayor que y menor que son algunos operadores relacionales comunes. Son la forma más común de establecer la condición en una sentencia if.

A continuación, se muestra una lista de todos los operadores relacionales que se pueden utilizar en las sentencias condicionales:

$x > y$: x es mayor que y

$x < y$: x es menor que y

$x \geq y$: x es mayor o igual que y

$x \leq y$: x es menor o igual que y

$x == y$: x es igual a y

$x != y$: x no es igual a y

Si la comparación da como resultado una sentencia verdadera, se ejecutará el cuerpo de la sentencia condicional.

○ Sentencias “if else”

Otra sentencia condicional útil es la sentencia if else. Así es como se escribe una sentencia if else:

```
if(condición) {
    }
else{
    }
```

Al igual que las sentencias if, las sentencias if-else evalúan primero la condición. Si la condición es verdadera, se ejecutará el código dentro del cuerpo del bloque if. Si la condición es falsa, se ejecutará el código del bloque else.

- **Sentencias “if else if”**

La sentencia if- else if permite comprobar dos condiciones. Este es el código de una sentencia if- else if:

```

If(condición 1){
}
else if(condición 2){
}
Else{
}

```

El programa comprueba primero la condición 1 en la sentencia if. Si es verdadera, se ejecuta el código en el cuerpo de la sentencia if, y luego el programa sale de todo el bloque if else if. Si la condición 1 es falsa, el programa comprueba la condición 2 en el bloque else if. Si la condición 2 es verdadera, se ejecuta el código dentro del cuerpo de la sección else if, luego el programa sale del bloque if- else if.

La sentencia if- else if es similar a tener dos sentencias if, pero hay una sutil diferencia. If else if sólo ejecuta un bloque – el bloque if o el bloque else if. El primer bloque que tiene una condición verdadera se ejecuta. Con dos sentencias if, ambas se ejecutan si las dos condiciones son verdaderas.

- **Bucles**

- **Bucle For.**

Este bucle también está en otros lenguajes de programación, pero en C ++ tiene una configuración muy flexible. Cuando se crea, el bucle toma tres «valores» (configuraciones): inicialización, condición y cambio. Un ciclo for generalmente contiene una variable que cambia a lo largo del ciclo, podemos usar su valor cambiante para nuestros propios fines. Una variable es local al bucle si se crea durante la inicialización.

```

for ( int i = 0; i < 100; i ++ ) {
    // cuerpo del ciclo
}

```

- **Operador break.**

Operador break (Inglés «break») le permite dejar el ciclo antes de lo programado, puede usarlo por condición o de cualquier manera conveniente. Por ejemplo, salir del ciclo antes de lo programado cuando se alcance un cierto valor:

```

for ( int i = 0; i < 100; i ++ ) {
    // abandona el bucle al llegar a 50
    if ( i == 50 ) break ;
}

```

➤ Operador continue («Continuar» en inglés)

Finaliza la iteración del ciclo actual antes de lo programado y pasa a la siguiente. Por ejemplo, completemos una matriz como lo hicimos anteriormente, pero omitamos un elemento:

```

// crea una matriz con 100 celdas
byte myArray [ 100 ] ;
for ( int i = 0; i < 100; i ++ ) {
    // si i es 10, omite
    if ( i == 10 ) continue ;

    // llena todas las celdas con 25
    myArray [ i ] = 25;
}

```

○ Bucle while.

Bucle while (Inglés «mientras»), también llamado ciclo con una condición previa, se ejecuta siempre que la condición especificada sea verdadera. Si la condición es inmediatamente incorrecta, el ciclo ni siquiera comenzará a funcionar y se omitirá por completo. Se declara de manera muy simple: la palabra clave while, luego la condición entre paréntesis, y ahora el cuerpo del ciclo:

```

int i = 0;
while ( i < 10 ) {
    i ++;
}

```

○ Bucle Do While.

Hacer mientras – «**do while**», el trabajo de este ciclo es completamente análogo al ciclo **while** con la excepción de que aquí la condición se establece después del ciclo, es decir **el ciclo se ejecutará una vez** como mínimo, luego verificará la condición, no al revés. Ejemplo:

```

do {
    // cuerpo del ciclo
} while ( condición ) ;

```

- **Funciones**

Una función es un conjunto de instrucciones que realizan una tarea específica. Además, puede ser diseñada e implementada de forma independiente al resto del código. Las funciones sirven principalmente para ordenar tu código y hacer que unas partes del programa sean independientes de otras. Esto te facilita mucho el trabajo cuando tienes que buscar y solucionar errores. Además, evitan repetir las mismas instrucciones de código una y otra vez.

En la imagen de abajo puedes ver cómo es la sintaxis de una función en Arduino y dos ejemplos reales. Al crear una función, lo primero que tienes que escribir es el tipo de dato que te va a devolver.

```
Tipo_variable function(variables){

    return algún_valor
}
```

Si por ejemplo quieres crear una función que mida la distancia, podrías utilizar tanto «float» como «double», para que te devuelva la distancia medida con decimales. En caso de que los decimales no te interesen, podrías utilizar «int». Si no necesitas que la función te devuelva algún dato, simplemente escribe «void». Ésto significa que no devuelve nada.

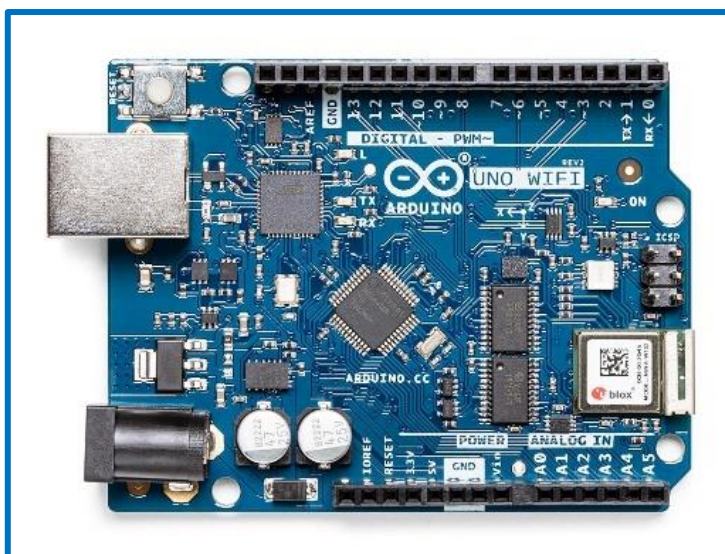
2. ENTORNO DE DESARROLLO PARA ARDUINO

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.



[Arduino mega](#)

Para poder entender este concepto, primero vas a tener que entender los conceptos de hardware y el software libres. El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas, pero igualmente funcionales al partir de la misma base.

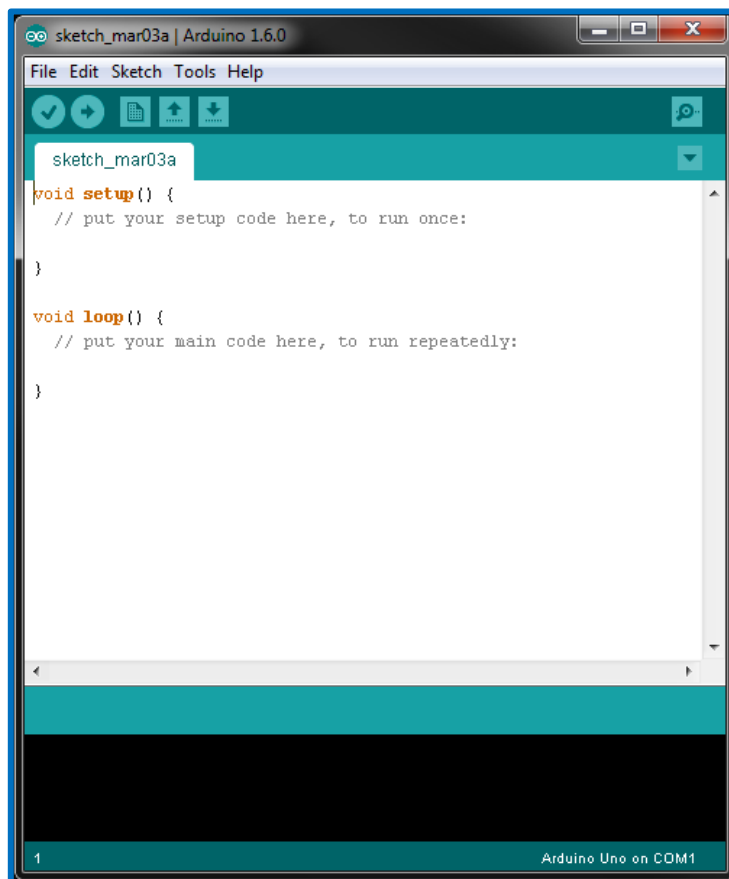


[Arduino de auditoría del fabricante.](#)

- **IDE – entorno de desarrollo integrado**

Llamado IDE por sus siglas en inglés de Integrated Development Environment, es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Además, en el caso de Arduino incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware a través del puerto serie.



Entorno IDE Arduino

Los programas de arduino están compuestos por un solo fichero con extensión “ino”, aunque es posible organizarlo en varios ficheros. El fichero principal siempre debe estar en una carpeta con el mismo nombre que el fichero. Anteriormente a la versión 1.x de Arduino se usaba la extensión “pde”. Cuando se pasó a la versión 1.x hubo grandes cambios, que deben tenerse en cuenta si se usa código antiguo.

La última versión del IDE de Arduino es la 1.6.8. Los grandes cambios del IDE Arduino se produjeron en el cambio de la versión 0.22 a la 1.0 y posteriormente en el cambio de la versión 1.0.6 a la 1.6.0 con grandes mejoras en el IDE de Arduino. El caso de la versión 1.6.0 los cambios han sido principalmente internos más que en el aspecto de la herramienta. También es destacable desde la aparición de la versión 1.6.2 la incorporación de la gestión de librerías y la gestión de placas, muy mejoradas respecto a la versiones anteriores y avisos de actualización de versiones de librerías y cores.

3. ENTRADAS Y SALIDAS DIGITALES Y ANALÓGICAS.

- **Entrada/Salida Digitales**

- **Función pinMode()**

Debe configurarse en la function setup(), esto indica como un pin dado tiene que comportarse como INPUT/OUTPUT. Ej. pinMode (pin, OUTPUT); configura el pin número 'pin' como de salida.

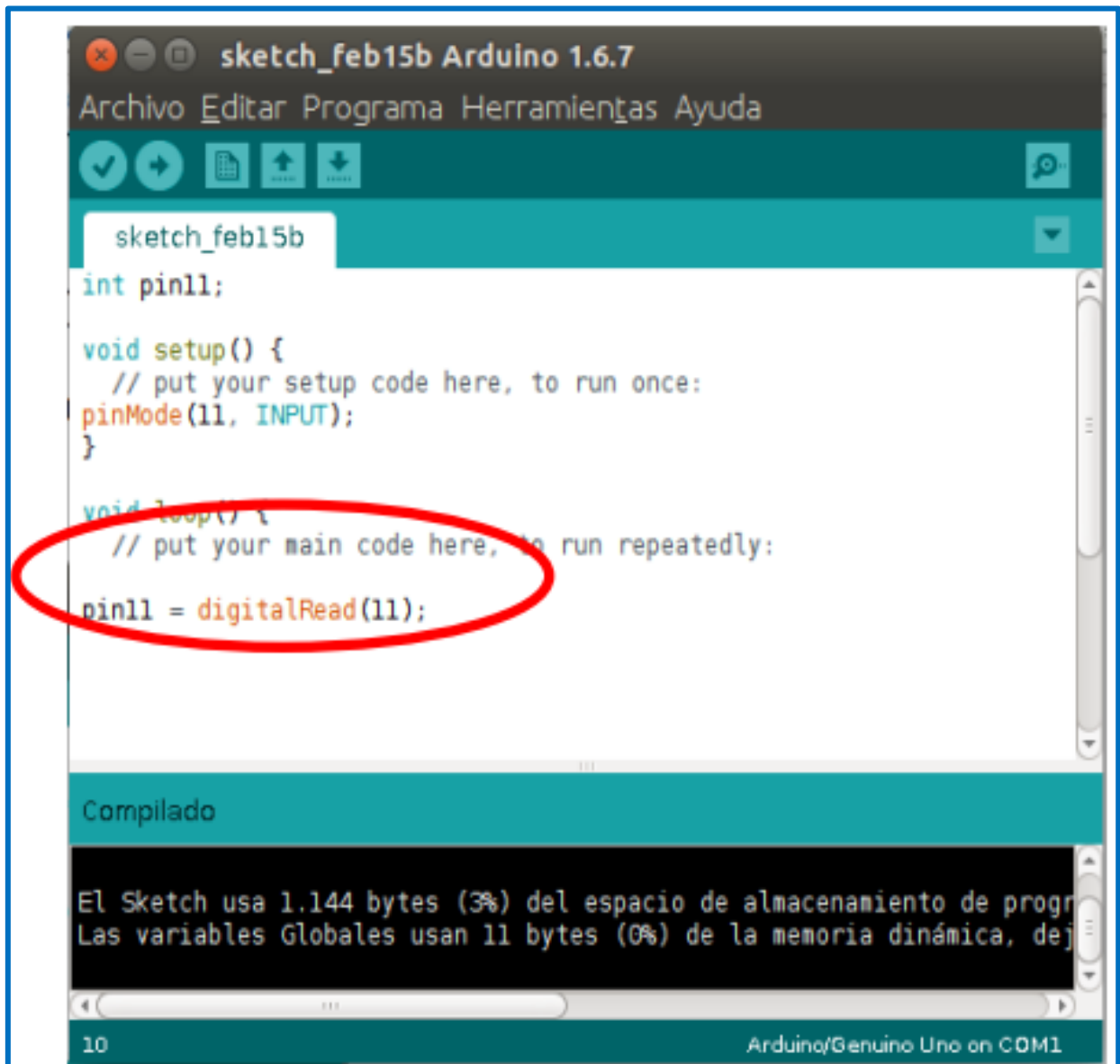
Los pines de Arduino funcionan por defecto como entradas, de forma que no necesitan declararse explícitamente como entradas empleando `pinMode()`. Se debe declarar el pin como variable y pasarlo a la función, o en se puede asignar directamente la variable, es de preferencia declarar los pines de entrada para llevar un orden a la hora de trabajar con el hardware.



[Función pinMode\(\)](#)

- **Función digitalRead()**

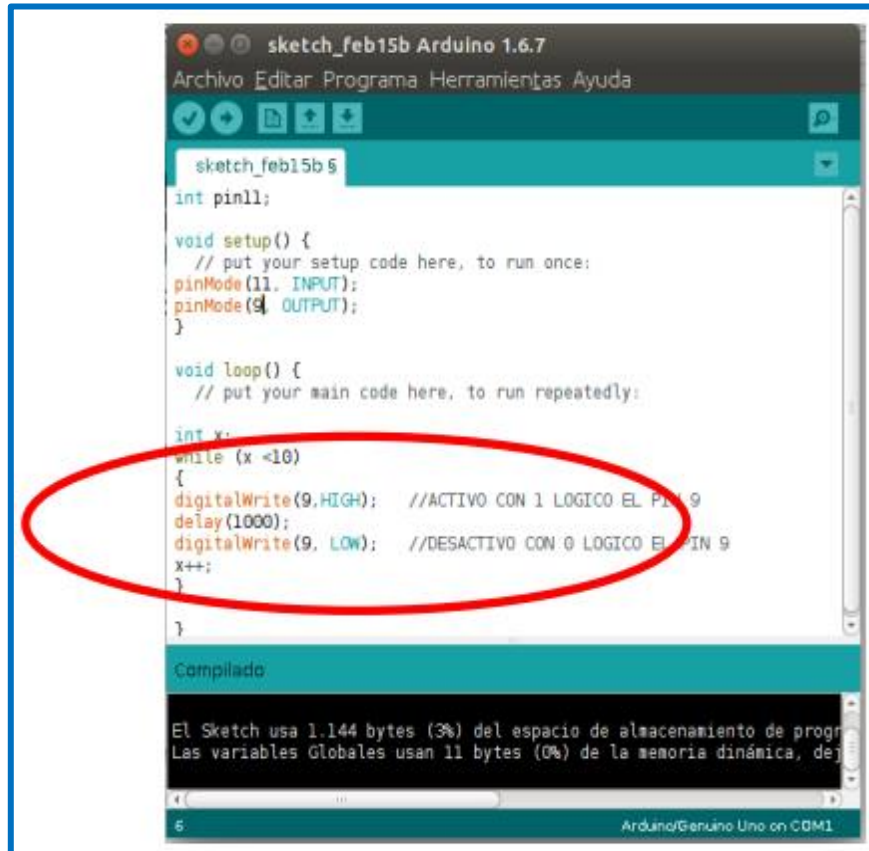
Lee el valor desde un pin digital específico. Devuelve un valor HIGH o LOW. El pin puede ser especificado con una variable o una constante (0-13). Ej. `pin11 = digitalRead(11);`



Función digitalRead()

- **Función digitalWrite()**

Introduce un nivel alto (HIGH) o bajo (LOW) en el pin digital especificado. De nuevo, el pin puede ser especificado con una variable o una constante 0-13. Ej. `digitalWrite(pin, HIGH);`



[Función digitalWrite\(\)](#)

○ Función analogRead(pin)

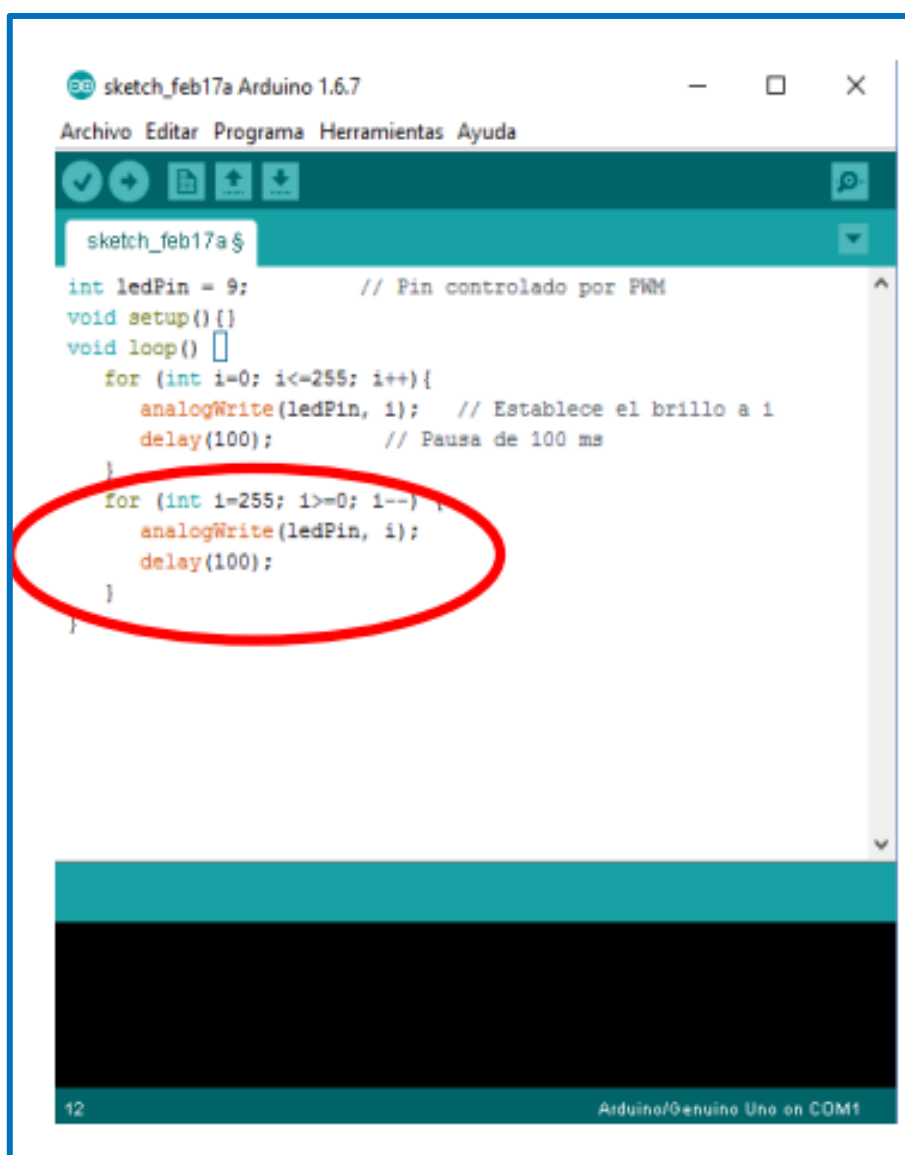
Lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT o OUTPUT.



[Función analogRead](#)

- **Función analogWrite()**

Escribe un valor pseudo-analógico usando modulación por ancho de pulso (PWM) en un pin de salida marcado como PWM. Ejemplo `analogWrite(pin, v);` // escribe 'v' en el 'pin' analógico. Puede especificarse un valor de 0 - 255. Un valor 0 genera 0 V en el pin especificado y 255 genera 5 V. Para valores de 0 a 255, el pin alterna rápidamente entre 0 V y 5 V, cuanto mayor sea el valor, más a menudo el pin se encuentra en HIGH (5 V). Por ejemplo, un valor de 64 será 0 V tres cuartas partes del tiempo y 5 V una cuarta parte. Un valor de 128 será 0 V la mitad del tiempo y 5 V la otra mitad. Un valor de 192 será 0 V una cuarta parte del tiempo y 5 V tres cuartas partes.



```
sketch_feb17a Arduino 1.6.7
Archivo Editar Programa Herramientas Ayuda
sketch_feb17a$
int ledPin = 9;          // Pin controlado por PWM
void setup() {}
void loop() {}
  for (int i=0; i<=255; i++){
    analogWrite(ledPin, i); // Establece el brillo a i
    delay(100);           // Pausa de 100 ms
  }
  for (int i=255; i>=0; i--){
    analogWrite(ledPin, i);
    delay(100);
  }
}
```

[Función analogWrite](#)

TAREA N°03

Usa sensores, actuadores y crea programas de adquisición de datos.

1. DEFINICIÓN DE SENSORES.

Hace referencia al dispositivo que proporciona una respuesta (normalmente mediante la generación de una señal eléctrica) frente a estímulos o señales físicas o químicas.

Al realizar una medida se toma energía del medio donde se mide, y éste se perturba. Por ello, la energía tomada debe ser mínima y la señal de salida (resultado de la transformación de la energía tomada del medio) es pequeña. El dominio eléctrico de las señales de salida permite la utilización de una gran cantidad de recursos electrónicos para su tratamiento (Cis amplificadores, filtros, etc.), transmisión o almacenamiento.

○ Clasificación de sensores

○ Según aporte de energía

- **Moduladores:** precisan una fuente externa de alimentación.
- **Generadores:** toman únicamente la energía del medio donde miden.

○ Según la señal de salida

- **Analógicos:** la salida varía de forma continua. Normalmente la información está en la amplitud. Cuando la información está en la frecuencia se denominan “cuasi-digitales”.
- **Digitales:** la salida varía en pasos discretos.

○ Según el modo de funcionamiento

- **Deflexión:** la magnitud medida genera un efecto físico (deflexión).
- **Comparación:** se intenta mantener nula la deflexión mediante la aplicación de un efecto opuesto al generado por la magnitud medida.

○ Según la relación

- Entrada-salida: orden cero, 1er orden, 2º orden

- **Según el principio físico**
 - Puede ser resistivo, capacitivo, termoelectrico o piezoeléctrico.
- **Según la magnitud media**
 - Se considera a la temperatura, la presión o la aceleración.

2. Definición de Actuadores.

Los actuadores son dispositivos que convierten energía (eléctrica, hidráulica, neumática, entre otras) en movimiento o fuerza mecánica. Se utilizan para controlar sistemas físicos como válvulas, motores, mecanismos robóticos, etc., en respuesta a una señal de control. Son fundamentales en sistemas de automatización, robótica y control industrial, ya que permiten la interacción física con el entorno tras recibir una señal de control de sensores o controladores.

- **Tipos de actuadores:**

- **Actuadores eléctricos**
 - **Motores eléctricos (DC/AC):** Convierten energía eléctrica en movimiento rotativo. Se usan en robots, electrodomésticos, ventiladores, y vehículos eléctricos.
 - **Servomotores:** Motores controlados por señales de retroalimentación (feedback) que permiten un control preciso de la posición angular.



[Servomotor MG996R engranaje 11KG](#)

- **Motores paso a paso (stepper motors):** Motores que giran en pasos discretos, permitiendo un control exacto de la posición y velocidad.
- **Solenoides:** Dispositivos que generan movimiento lineal cuando se aplica corriente, usados en válvulas, cerraduras electrónicas, y relevadores.



[Solenoides de arranque](#)

○ **Actuadores neumáticos**

- Utilizan aire comprimido para generar movimiento lineal o rotativo.
- **Cilindros neumáticos:** Producen un movimiento lineal al permitir el paso de aire en una o ambas direcciones.
- **Motores neumáticos:** Convierte la presión del aire en movimiento rotativo. Se usan donde es necesario evitar el uso de electricidad, como en entornos inflamables.
- **Ventajas:** Son rápidos y suelen ser más seguros que los actuadores eléctricos en ambientes explosivos.

○ **Actuadores hidráulicos**

- Funcionan mediante la presión de un fluido hidráulico (generalmente aceite) para generar movimiento.
- **Cilindros hidráulicos:** Producen un movimiento lineal muy potente. Se usan en maquinaria pesada, como excavadoras y grúas.

- **Motores hidráulicos:** Transforman la presión de un fluido en movimiento rotativo.
- **Ventajas:** Proporcionan gran fuerza y precisión en aplicaciones de alto rendimiento.

- **Actuadores térmicos**

- **Bimetálicos:** Utilizan la expansión térmica de metales para producir movimiento. Comúnmente usados en termostatos y algunos interruptores automáticos.
- **Cera térmica:** Utilizan la expansión de materiales sensibles al calor para generar fuerza. Son usados en válvulas termostáticas de sistemas de refrigeración o calefacción.

- **Actuadores piezoeléctricos**

Convierten una señal eléctrica en una vibración o movimiento mecánico. Utilizan materiales que se deforman cuando se les aplica una corriente eléctrica.

- **Usos:** En sistemas de precisión, como en impresoras de inyección de tinta, relojes, y sistemas de posicionamiento de alta precisión.

- **Actuadores magnéticos**

Funcionan mediante campos magnéticos para producir movimiento.

- **Actuadores electromagnéticos:** Son solenoides mejorados que permiten el movimiento a través de fuerzas magnéticas más controladas.
- **Usos:** En dispositivos de levitación magnética, trenes de levitación y algunos tipos de robots.

- **Actuadores mecánicos**

- **Levas y bielas:** Transforman un movimiento rotativo en uno lineal, o viceversa.
- **Engranajes y poleas:** Usados para multiplicar o distribuir la fuerza o el movimiento de un motor o fuerza aplicada.

- **Clasificación según el tipo de movimiento:**

- **Lineales:** Producen movimiento recto, como en los cilindros neumáticos,

hidráulicos y algunos solenoides.

- **Rotativos:** Producen un movimiento giratorio, como en los motores eléctricos, hidráulicos y neumáticos.

3. DEFINICIÓN DE MICROCONTROLADORES.

Nos indica Gunther Gridling que Un microcontrolador (abreviado μC , UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Donde podemos concluir que un microcontrolador es aquel dispositivo que se encarga de controlar los procesos para el cual ha sido diseñado en su programación.

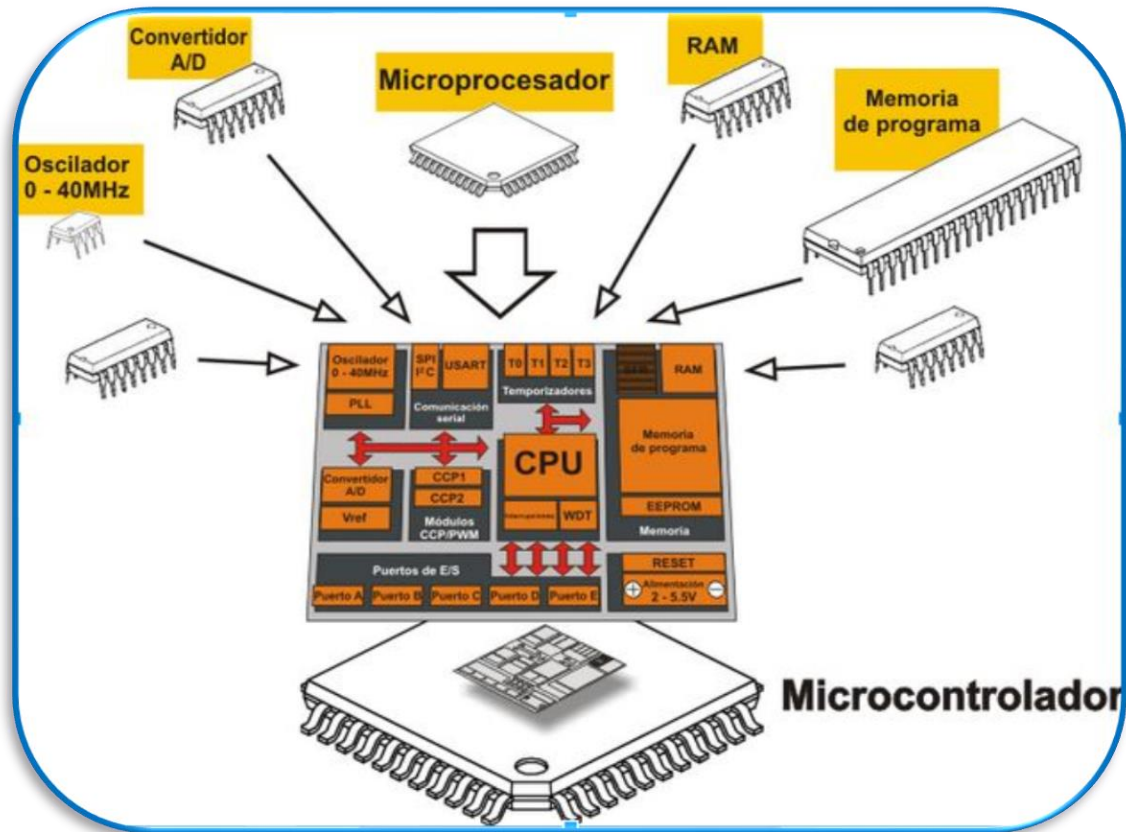
• Características

Algunos microcontroladores se caracterizan por poder utilizar 4 bits (ejemplo 1010), y funcionan a velocidad de reloj con frecuencias muy bajas como por ejemplo a 4 kHz, ocasionando un pequeño consumo donde la potencia estaría alrededor de los milivatios (mW).

El microcontrolador puede mantenerse en espera de un evento detectando mediante sus periféricos algún evento de cambio; el consumo de energía del microcontrolador en espera (reloj de la CPU y los periféricos de la mayoría) es muy pequeño casi siempre está alrededor de los nanovatios, lo que hace que ideal para trabajos donde se tiene una batería dando una larga duración debido a su poco consumo.

Un microcontrolador, no contiene ninguna información, su memoria ROM está vacía en la espera de datos. Para que pueda controlar uno o varios procesos es indispensable crear algún programa en lenguaje ensamblador para ser grabado en la memoria EEPROM del controlador y así poder usarlo en los procesos para el cual ha sido diseñado; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador en la EEPROM, debe ser compilado en sistema numérico hexadecimal que es el sistema que hace funcionar al microcontrolador cuando se ponga en marcha con el resto de sus periféricos de entrada/salida.

El procesador es el parte principal del sistema es el núcleo donde se encarga de analizar y procesar toda la información que pasa por él, entonces cuanto más rápido sea nuestro procesador más rápido ejecutara las instrucciones que tiene contenido en la memoria.



[El microcontrolador](#)

Según Jose Adolfo González nos indica que Los Procesadores están compuestos por circuitos que realizan operaciones lógicas y matemáticas, para dicho trabajo se le asigna una unidad completa. Es aquí donde se realizan las operaciones de suma, resta y el álgebra de Boole.

En la actualidad han mejorado su rendimiento y los procesadores ya incorporan una o más ALU, algunas ALU se encargan de realizar operaciones complejas, como en coma flotante, de acuerdo a su trabajo se le otorgado un nombre como por ejemplo el coprocesador matemático.

Su impacto en las prestaciones del procesador es también importante porque, dependiendo de su potencia, tareas más o menos complejas, pueden hacerse en tiempos muy cortos.

- **Registros**

Son pequeños espacios de memorias necesarios para un microprocesador, debido que estos espacios de memoria sirven para almacenar los datos para las operaciones que deben realizar los circuitos del procesador. Los registros almacenan los resultados de las instrucciones, que son cargados desde la memoria externa o en ella misma.

Nos indica Gunther Gridling que una parte de los registros está destinada a los datos, es la que determina uno de los parámetros más importantes de cualquier microprocesador. Cuando escuchamos que un procesador es de

4, 8, 16, 32, 64 o 128 bits, nos estamos refiriendo a procesadores que realizan sus operaciones con registros de datos de ese tamaño, y por supuesto, esto determina muchas de las potencialidades de máquinas.

A mayor número de bits de los registros de datos del procesador, mayores serán las posibilidades para los datos, en cuanto a poder de cómputo y velocidad de ejecución, ya que este parámetro determina la potencia que se puede incorporar al resto de los componentes del sistema.

- **Unidad de control**

Nos indica Jared Tirado Martínez que la unidad de control es la más importante en el procesador, en ella recae la lógica necesaria para la decodificación y ejecución de las instrucciones, el control de los registros, la ALU, los buses y cuanto cosa más se quiera meter en el procesador. La unidad de control es una parte fundamental que determina el potencial del procesador, debido a que su estructura determina sus parámetros como el conjunto de instrucciones que pueda ejecutar, velocidad de ejecución, frecuencia de la máquina, el bus que puede tener el sistema, manejo de interrupciones.

Las unidades de control son el elemento más complejo de un procesador y por lo general está dividido en unidades más pequeñas todas trabajadas como una sola unidad. La unidad de control es un conjunto de componentes, la unidad de decodificación, unidad de ejecución, controladores de memoria cache, controladores de buses, controlador de instrucciones, pipelines, entre otros elementos, dependiendo siempre del tipo de procesador.

- **Buses**

Es el medio por el cual se comunican los distintos componentes del procesador para interactuar entre sí, eventualmente los buses o una parte de ellos estarán reflejados en los pines del encapsulado del procesador.

En los microcontroladores, no es normal que los buses estén reflejados en el encapsulado del circuito, debido a que estos se destinan principalmente a las E/S de propósito general y periféricos del sistema.

Existen tres tipos de buses:

- **Dirección:** Se utiliza para seleccionar al dispositivo con el cual se quiere trabajar o en el caso de las memorias, seleccionar el dato que se desea leer o escribir.
- **Datos:** Se utiliza para mover los datos entre los dispositivos de hardware (entrada y salida).
- **Control:** Se utiliza para gestionar los distintos procesos de escritura/lectura y controlar la operación de los dispositivos del sistema.

- **Conjunto de instrucciones**

Es el grupo de líneas de comando (Instrucciones) que puede realizar el procesador, limitando en sus acciones debidas que las instrucciones son limitadas por cada modelo de procesador. O. Bustillo Olivas y C. Martínez Hernández define que las operaciones básicas que puede realizar el procesador, que conjugadas y organizadas forman lo que conocemos como software. El conjunto de instrucciones, vienen siendo como las letras del alfabeto, el elemento básico del lenguaje, que organizadas adecuadamente permiten escribir palabras, oraciones y cuanto programa se le ocurra.

Existen dos tipos básicos de almacenaje de instrucciones, que determinan la arquitectura del procesador estos son, el conjunto de instrucciones complejas (CISC) y Conjunto de Instrucciones Reducidas (RISC). El CISC, del inglés Complex instruction set Computing, Computadora de Conjunto de Instrucciones Complejo. Los microprocesadores CISC tienen un conjunto de instrucciones que se caracteriza por ser muy amplio y que permiten realizar operaciones complejas entre operandos situados en la memoria o en los registros internos.

Este tipo de repertorio dificulta el paralelismo entre instrucciones, por lo que en la actualidad, la mayoría de los sistemas CISC de alto rendimiento convierten las instrucciones complejas en varias instrucciones simples del tipo RISC, llamadas generalmente microinstrucciones.

4. CIRCUITO BÁSICO ELECTRÓNICO Y LEY DE OHM.

Un circuito electrónico es un conjunto de componentes eléctricos interconectados para realizar una función específica, como amplificar señales, procesar datos o controlar dispositivos. Los circuitos electrónicos están presentes en casi todos los dispositivos modernos, desde teléfonos móviles hasta computadoras y electrodomésticos. Estos circuitos funcionan permitiendo que la corriente eléctrica fluya a través de componentes como resistencias, condensadores, transistores, diodos y otros elementos electrónicos.

Los componentes principales de un circuito electrónico son:

- **Resistencias:** Limitan el flujo de corriente en el circuito. Se utilizan para proteger otros componentes o dividir tensiones.
- **Condensadores:** Almacenan y liberan energía eléctrica. Se usan para filtrar señales, estabilizar voltajes y temporizar circuitos.
- **Diodos:** Permiten el paso de la corriente en una sola dirección. Se utilizan en rectificadores, protección de circuitos y en la construcción de LED.

- **Transistores:** Funcionan como interruptores o amplificadores. Son componentes clave en la electrónica digital y analógica, siendo la base de los procesadores y circuitos de amplificación.
- **Bobinas (inductores):** Almacenan energía en forma de campo magnético. Son usados en circuitos de filtrado, fuentes de alimentación, y radios.
- **Circuitos integrados (ICs):** Chips que contienen varios componentes electrónicos miniaturizados. Pueden realizar tareas muy específicas, como procesar señales de audio, controlar dispositivos o gestionar datos en un microprocesador.
- **Conectores y cables:** Permiten la conexión física entre diferentes partes del circuito o entre dispositivos.

La clasificación de los circuitos electrónicos es:

- **Circuitos analógicos**

Trabajan con señales continuas (analógicas), como las señales de audio o de video. Ejemplo: Amplificadores de audio que aumentan la potencia de una señal para que pueda ser reproducida por un altavoz. Los componentes comunes son resistencias, condensadores, transistores, amplificadores operacionales.

- **Circuitos digitales**

Procesan señales discretas, es decir, señales que solo pueden tener dos estados: alto (1) o bajo (0). Esto es la base del sistema binario, usado en computadoras y electrónica digital. Ejemplo: Microprocesadores y microcontroladores, que realizan cálculos y toman decisiones en función de entradas digitales. Los componentes comunes son compuertas lógicas, flip-flops, registros, contadores y memorias.

- **Circuitos mixtos (analógicos y digitales)**

Combina características de circuitos analógicos y digitales en un mismo diseño. Son comunes en dispositivos que interactúan con el mundo real (analógico) pero procesan la información de forma digital.

Ejemplo: Convertidores analógico-digital (ADC) y digital-analógico (DAC), que convierten señales de un tipo al otro.

Todos los circuitos se pueden componer en tres elementos básicos:

- **Fuente de voltaje:** es la que da la corriente al circuito electrónico. También se dicen fuentes de alimentación.

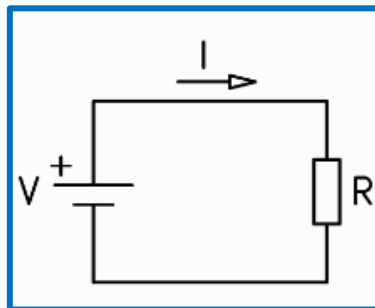
- **Carga:** es un dispositivo que consume energía eléctrica en forma de corriente y la transforma en otras formas. Sin la carga, no tiene mucho sentido tener un circuito. En circuitos complejos, la carga es una combinación de componentes, como resistencias, condensadores, transistores, etc.
- **Trayectoria conductora:** proporciona una ruta a través del cual fluye la corriente. Esta ruta comienza en la fuente de voltaje, viaja a través de la carga y luego regresa a la fuente de voltaje. Esta ruta debe formar un bucle desde el lado negativo de la fuente de voltaje al lado positivo de la fuente de voltaje.



[Circuito electrónico](#)

- **Ley de ohm**

La ley de Ohm es una ley física que relaciona, en un circuito eléctrico, el valor de la intensidad de corriente, la tensión aplicada y la resistencia eléctrica del circuito.



[Ley de ohm](#)

La ley de Ohm relaciona tres magnitudes eléctricas que están representadas en tres unidades.

- **Tensión o Voltaje:**

Es la energía con la que una pila impulsa a los electrones a través del circuito. Cuando el voltaje es más alto, la energía es mayor y por lo tanto

los electrones circulan más deprisa, aumentando la corriente eléctrica. La tensión o voltaje se mide en Voltios.

- **Resistencia Eléctrica:**

Es la oposición que presenta un componente al paso de la corriente eléctrica. Cuanto mayor es la resistencia, más se opone al paso de la corriente eléctrica y por lo tanto menos corriente pasará a través del circuito. La resistencia eléctrica se mide en Ohmios.

- **Intensidad de Corriente:**

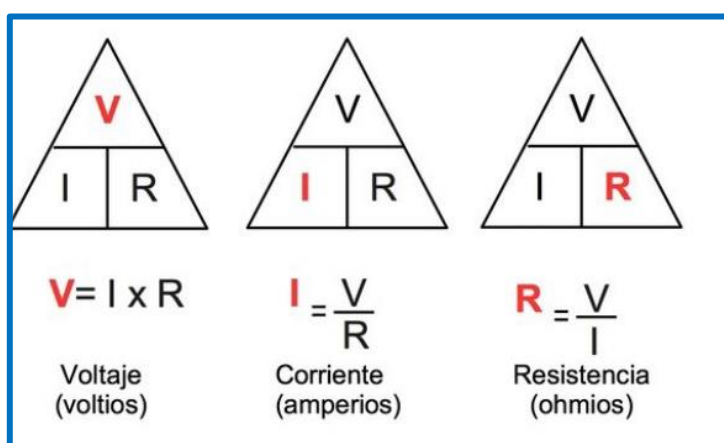
Es la cantidad de electrones que circulan por un conductor cada segundo. Cuantos más electrones circulen por segundo, más corriente pasará por el conductor. La intensidad de corriente se mide en Amperios.

- **Fórmula de la ley de Ohm**

Según la ley de Ohm la intensidad de corriente (I) que circula por una resistencia eléctrica (R) es proporcional a la tensión (V) aplicada a la resistencia e inversamente proporcional al valor de la resistencia eléctrica.

La ley de Ohm escrita en notación matemática es la siguiente:

$$resistencia = \frac{voltaje}{intensidad\ de\ corriente}$$



[Ley de Ohm](#)

5. Señales digitales y analógicas.

- **Conceptos**

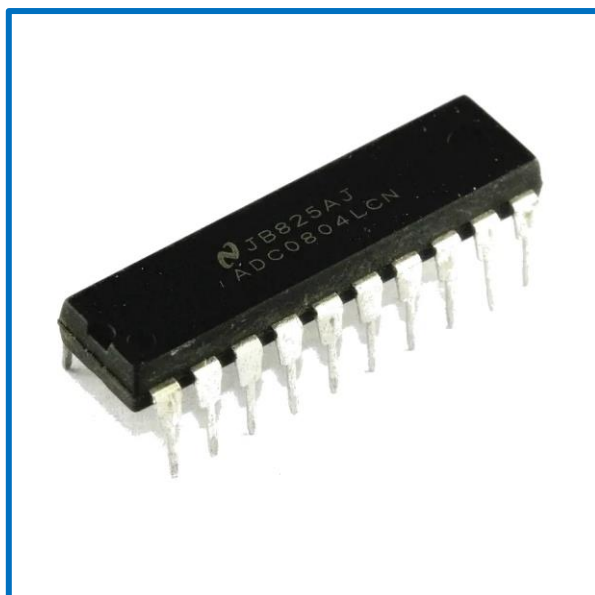
- **Señal Analógica:** Es una señal continua que varía en el tiempo y puede tomar infinitos valores dentro de un rango específico. Representa fenómenos físicos como temperatura, presión, velocidad o sonido, que

son también continuos. Estas señales suelen tener forma de ondas senoidales o curvas suaves y se caracterizan por tener amplitud y frecuencia.

- **Señal Digital:** Es una señal discreta que solo puede tomar un número limitado de valores, comúnmente dos: alto (1) y bajo (0), lo que representa el sistema binario de la electrónica digital. Las señales digitales son ideales para los sistemas de control y procesamiento en dispositivos IoT, ya que pueden ser procesadas por microcontroladores y computadoras de manera eficiente.
- **Diferencias entre Señales Digitales y Analógicas**
 - **Continuidad:**
 - **Analógica:** Continuas y pueden tener cualquier valor dentro de su rango.
 - **Digital:** Discretas, solo pueden ser 0 o 1 (en el caso de binario).
 - **Susceptibilidad a Interferencias:**
 - **Analógica:** Sensible al ruido y las interferencias. Esto puede distorsionar la señal.
 - **Digital:** Menos susceptible al ruido; cualquier distorsión se ignora mientras se mantenga dentro de ciertos límites.
 - **Calidad de Transmisión:**
 - **Analógica:** La calidad se puede degradar en largas distancias.
 - **Digital:** Se puede regenerar la señal en cada transmisión, manteniendo la calidad.
 - **Procesamiento:**
 - **Analógica:** Requiere procesamiento con circuitos que interpreten las variaciones de voltaje o corriente.
 - **Digital:** Se procesa en sistemas digitales, que interpretan los valores binarios.
- **Uso de Señales en Sensores y Actuadores para IoT**
 - **Sensores Analógicos:** Como sensores de temperatura, luz, presión, etc., miden valores que varían continuamente. La señal de salida suele

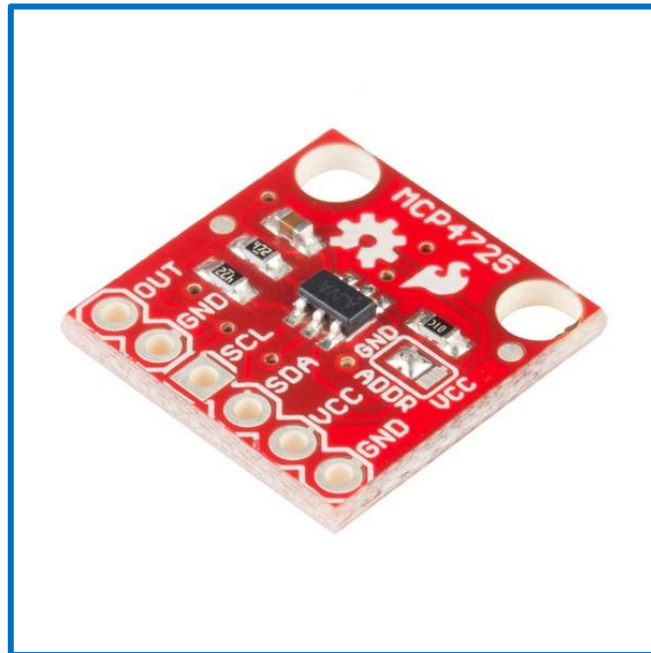
ser un voltaje o corriente proporcional al fenómeno medido.

- **Sensores Digitales:** Producen una salida binaria, que puede indicar estados específicos (como encendido o apagado). Los sensores digitales son ideales para aplicaciones donde solo se necesitan dos estados, como interruptores o detectores de movimiento.
 - **Actuadores:** Los actuadores pueden ser controlados mediante señales digitales (por ejemplo, un motor que enciende o apaga) o analógicas (como un motor de velocidad variable que responde a voltajes diferentes).
- **Conversión entre Señales Analógicas y Digitales**
 - **Conversor ADC (Analog-to-Digital Converter):** Convierte una señal analógica en una señal digital para que pueda ser procesada por un microcontrolador. Por ejemplo, si un sensor de temperatura entrega un voltaje variable, el ADC lo convierte a un valor digital que el sistema IoT puede interpretar.



[ADC0804 Convertidor Analógico-Digital](#)

- **Conversor DAC (Digital-to-Analog Converter):** Convierte una señal digital en analógica. Es útil para sistemas donde la salida deba variar de manera continua, como en el control de volumen de un altavoz o el ajuste de una luz.



[Conversor DAC MCP4725 Sparkfun BOB-12918](#)

• Aplicación en Sistemas IoT

- En sistemas IoT, los datos de sensores suelen ser convertidos a formato digital para ser procesados y transmitidos a través de redes. Las señales analógicas pueden capturar variaciones detalladas del entorno, mientras que las digitales permiten el procesamiento eficiente y la comunicación confiable en redes IoT.
- La integración de sensores y actuadores que funcionan con ambos tipos de señales es esencial para una solución IoT completa, permitiendo capturar datos precisos y realizar acciones basadas en esos datos.

• Ejemplo Práctico de Señales en IoT

Imaginemos un sistema IoT para el monitoreo ambiental:

- **Sensores analógicos:** Miden temperatura y humedad y envían señales analógicas que representan estos valores.
- **Conversión:** Un ADC en el microcontrolador convierte estas señales analógicas en datos digitales para el procesamiento.
- **Actuador digital:** Cuando la temperatura supera un umbral, se activa un ventilador (controlado digitalmente) para ventilar el ambiente.

Esta combinación de señales permite un control preciso y eficiente, clave para los sistemas IoT que requieren tanto medición continua como respuestas discretas.

TAREA N°04**Revisa la comunicación y conectividad IoT.****1. ¿QUÉ ES RASPBERRY PI?**

La Raspberry Pi es una computadora de placa única (SBC, por sus siglas en inglés) desarrollada por la Fundación Raspberry Pi en el Reino Unido. Está diseñada para ser económica, compacta y fácil de usar, orientada a fomentar el aprendizaje de la informática y la programación en diversas áreas, incluyendo proyectos de IoT, robótica, y automatización. Ofrece capacidades de computación completas, lo que la convierte en una excelente opción para proyectos que requieren conectividad, procesamiento de datos, y control de dispositivos.

En IoT, la Raspberry Pi se utiliza principalmente como un controlador y centro de procesamiento de datos, integrándose con sensores y actuadores para recopilar, analizar, y enviar datos a través de internet.

2. MODELOS DE PLACAS.

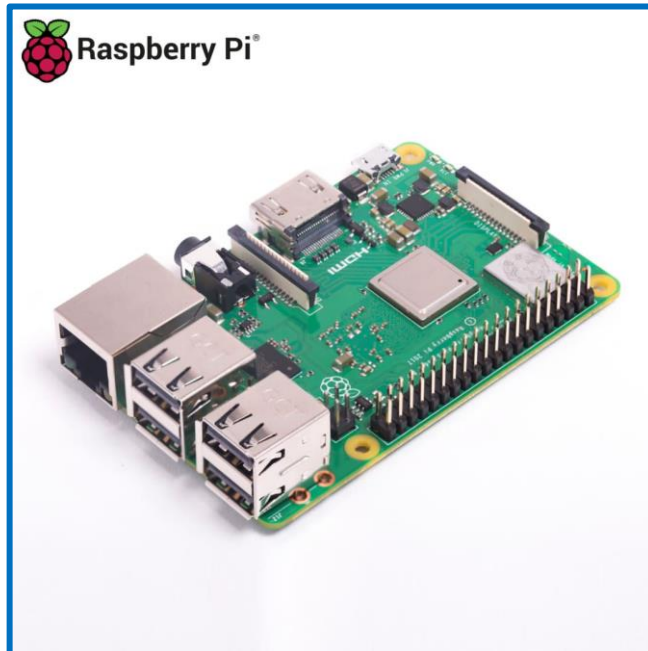
Existen varios modelos de Raspberry Pi, cada uno con características y precios diferentes:

- **Raspberry Pi 4:** La versión más avanzada y potente, disponible con 2GB, 4GB o 8GB de RAM. Tiene un procesador más rápido y soporte para dos pantallas 4K.



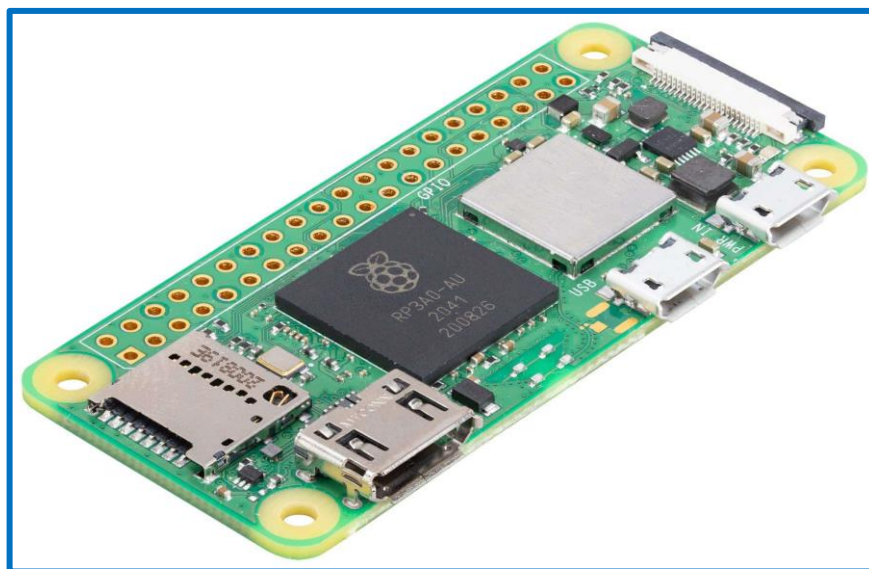
[Raspberry Pi 4B de 4GB](#)

- **Raspberry Pi 3:** Un modelo popular, con soporte WiFi y Bluetooth, 1GB de RAM, y un procesador de 1.2 GHz. Ideal para proyectos IoT de mediana complejidad.



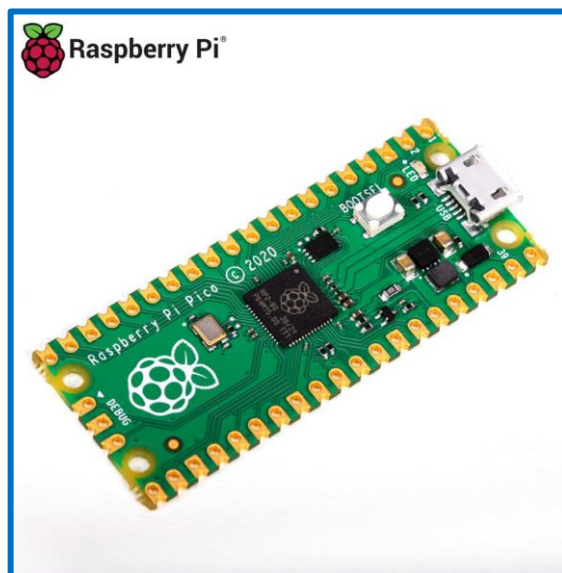
[Raspberry Pi 3B+](#)

- **Raspberry Pi Zero y Zero W:** Son versiones más compactas y de bajo costo, pero con menos potencia. La versión Zero W incluye WiFi y Bluetooth, útil para proyectos IoT portátiles.



[Raspberry Pi Zero 2W](#)

- **Raspberry Pi Pico:** Una placa basada en microcontrolador diseñada para proyectos que requieren control de hardware y bajo consumo energético. No incluye capacidades de procesamiento de sistemas operativos completos como las demás.



[Raspberry Pi Pico](#)

3. PARTES PRINCIPALES DE LA PLACA.

Las partes esenciales de una Raspberry Pi incluyen:

- **Puerto HDMI:** Para conectar la placa a una pantalla.
- **CPU (Procesador):** La unidad de procesamiento central de la placa, que ejecuta el sistema operativo y los programas.
- **Memoria RAM:** Para almacenamiento temporal de datos durante la ejecución de programas.
- **Puertos GPIO (General Purpose Input/Output):** Pines que permiten la conexión directa con sensores, actuadores y otros dispositivos.
- **Almacenamiento (MicroSD):** La Raspberry Pi utiliza una tarjeta microSD para almacenar el sistema operativo y los datos.
- **Puertos USB:** Permiten la conexión de dispositivos como teclados, ratones y almacenamiento USB.
- **Puerto HDMI:** Para conectar la placa a una pantalla.
- **Conectividad:** Algunos modelos incluyen WiFi, Bluetooth y Ethernet, lo que facilita la conexión a redes y dispositivos.

4. SISTEMAS OPERATIVOS PARA RASBERRY PI.

La Raspberry Pi es compatible con varios sistemas operativos, aunque Raspberry Pi OS (anteriormente Raspbian) es el oficial y más utilizado. Otros sistemas populares incluyen:

- **Raspberry Pi OS:** Basado en Debian Linux, optimizado para la Raspberry Pi. Incluye herramientas educativas y de desarrollo.
- **Ubuntu:** Canonical ofrece una versión específica de Ubuntu para Raspberry Pi, ideal para proyectos que requieren un sistema operativo robusto y de uso empresarial.
- **Kali Linux:** Sistema operativo orientado a la seguridad informática y pruebas de penetración.
- **Windows IoT Core:** Una versión ligera de Windows orientada a dispositivos IoT.
- **LibreELEC y OSMC:** Para transformar la Raspberry Pi en un centro multimedia.

5. PROTOCOLOS HTTP, TCP, IP, MQTT, LORAWAN, ETC.

En IoT, se utilizan varios protocolos de comunicación para conectar dispositivos y transmitir datos de manera eficiente:

- **HTTP (HyperText Transfer Protocol):** Un protocolo para la comunicación web. Es fácil de implementar, pero no tan eficiente para dispositivos IoT de bajo consumo y alto volumen de datos.



[Protocolo HTTP](#)

- **TCP/IP (Transmission Control Protocol/Internet Protocol):** El conjunto de protocolos que permite la comunicación en redes y la base para la mayoría de conexiones en Internet.



[TCP/IP](#)

- **MQTT (Message Queuing Telemetry Transport):** Protocolo ligero y eficiente para la comunicación IoT, especialmente en dispositivos con limitaciones de ancho de banda y potencia. Utiliza un modelo de publicación/suscripción.



[MQTT para comunicación de dispositivos](#)

- **LoRaWAN (Long Range Wide Area Network):** Protocolo para redes de largo alcance y bajo consumo. Ideal para dispositivos IoT que envían datos a distancias largas sin consumir mucha energía.



[LoRaWAN](#)

- **CoAP (Constrained Application Protocol):** Protocolo ligero que funciona sobre UDP y es adecuado para dispositivos con recursos limitados en redes IoT.

6. VENTAJAS DE NODEMCU ESP8266 SOBRE ARDUINO UNO.

El NodeMCU ESP8266 y el Arduino Uno son dos placas populares en proyectos de IoT, pero el ESP8266 tiene varias ventajas sobre el Arduino Uno en este contexto:

- **Conectividad WiFi integrada:** El ESP8266 incluye un módulo WiFi, lo que facilita la conexión a redes y evita la necesidad de módulos externos. El Arduino Uno no tiene WiFi de manera predeterminada.
- **Mayor velocidad y memoria:** El ESP8266 tiene una velocidad de reloj de hasta 80 MHz y más memoria en comparación con el Arduino Uno, lo que permite ejecutar programas más complejos.
- **Bajo costo:** El ESP8266 es generalmente más económico que una configuración Arduino Uno con un módulo WiFi adicional.
- **Facilidad de programación:** Aunque ambos pueden ser programados en el entorno de Arduino IDE, el ESP8266 también permite el uso de otros lenguajes y herramientas de desarrollo, como MicroPython y Lua, que son útiles para desarrolladores con experiencia en esos lenguajes.
- **Consumo de energía:** El ESP8266 está diseñado para un consumo bajo, lo que lo hace ideal para proyectos IoT que requieren autonomía energética.

REFERENCIAS BIBLIOGRÁFICAS

- *Algoritmo y Diagramas de flujo.* (s/f). Feriadetecnologia.com. http://www.feriadetecnologia.com/arduino/algoritmo_y_diagramas_de_flujo.html
- Componentes electrónicos: Los más utilizados. (2020, abril 14). *Electronic Board.* <https://www.electronicboard.es/componentes-electronicos-los-mas-utilizados/>
- Fromaget, P. (s/f). *Los 15 Mejores Sistemas Operativos Para Tu Raspberry Pi.* Raspberrytips.es. <https://raspberrytips.es/mejores-so-raspberry-pi/>
- Gandica, K. Y. M. (2018, junio 25). Unidad II Fundamentos del Harware Libre. *Blogspot.com.* <http://softwarelibreupt.blogspot.com/p/unidad-ii.html>
- *Guía de Referencia de Arduino.* (s/f). Arduino.cc. <https://www.arduino.cc/reference/es/>
- Microcontrolador - qué es y para que sirve. (2017, noviembre 12). *HeTPro-Tutoriales.* <https://hetpro-store.com/TUTORIALES/microcontrolador/>
- *¿Qué es un sensor y qué hace?* (s/f). Soluciones de Adquisición de Datos (DAQ). <https://dewesoft.com/es/blog/que-es-un-sensor>



RDA
RECURSO DIDÁCTICO PARA EL APRENDIZAJE