Módulo 1: Introducción. Conceptos

Presentación del IDE y de la placa.

- ¿Qué es Arduino? Como instalar, componentes, seleccionar placa
- variables y tipos de datos.
- funciones básicas
- esquemático, entradas salidas donde conectar
  led, pulsador, potenciómetro, ldr.
- protoboard.

Introducción a la electrónica básica.

- ley ohm
- conexión de led - cálculos
- pulsadores resistencia de pull
- anti-rebote hardware y software

Práctica, encendido de un led con pulsador pulsador.

- simuladores
- Puerto serie
- práctica

MATERIALES: (pulsadores, Leds,)

Ejercicios:

1) Blink Led
2) hola mundo presionando un botón
3) OPCIONAL enviar por serie un caracter y que haga algo
4) OPCIONAL toggle led con boton (opcional ver debounce)
5) Con 2 botones subir y bajar una variable. ej. velocidad de parpadeo
6) Con 2 botones seleccionar led y prenderlo
7) Comunicar al menos 2 placas entre sí.

¿Qué es Arduino?

Arduino es una plataforma electrónica de código abierto basada en hardware y software fácil de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida: activar un motor, encender un LED, publicar algo en línea. Puedes decirle a su placa qué hacer enviando un conjunto de instrucciones al microcontrolador en la placa. Para ello se utiliza el lenguaje de programación Arduino (basado en Wiring), y el Software Arduino (IDE), basado en Processing.

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta instrumentos científicos complejos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto, sus contribuciones se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos.

Arduino nació en el Instituto de Diseño de Interacción Ivrea como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta de simples placas de 8 bits a productos para aplicaciones IoT, wearables, impresión 3D y entornos integrados.
(según la pagina oficial https://www.arduino.cc/en/Guide/Introduction)

¿Qué es un IDE?

Un IDE (Entorno de Desarrollo Integrado, por sus siglas en inglés) es una herramienta de software que proporciona un conjunto de características y herramientas para facilitar el desarrollo de software. Un IDE generalmente incluye un editor de código, un compilador, un depurador y otras utilidades que ayudan a los programadores a escribir, probar y depurar su código de manera más eficiente.

Las principales características de un IDE son las siguientes:

1. Editor de código: Proporciona un entorno para escribir y editar el código fuente de manera cómoda, con características como resaltado de sintaxis, autocompletado y funciones de búsqueda y reemplazo.
2. Compilador o intérprete: Permite convertir el código fuente en un programa ejecutable o en instrucciones interpretables. El compilador traduce el código a lenguaje de máquina, mientras que el intérprete ejecuta el código línea por línea.
3. Depurador: Ayuda a identificar y corregir errores en el código durante el proceso de desarrollo. Permite establecer puntos de interrupción, inspeccionar variables y ejecutar el código paso a paso para encontrar y solucionar problemas.
4. Gestión de proyectos: Permite organizar y administrar los archivos del proyecto, como el código fuente, bibliotecas y otros recursos. También proporciona herramientas para compilar, ejecutar y empaquetar el proyecto.

El IDE de Arduino es una aplicación de software que incluye todas estas características y está específicamente diseñado para programar y desarrollar proyectos con placas Arduino. Proporciona una interfaz amigable y simplificada, así como librerías y ejemplos predefinidos que facilitan el proceso de programación en Arduino.

Descargar el Arduino ide
para descargar el IDE de Arduino, puedes dirigirte al sitio web oficial de Arduino en la siguiente dirección:

https://www.arduino.cc/en/software

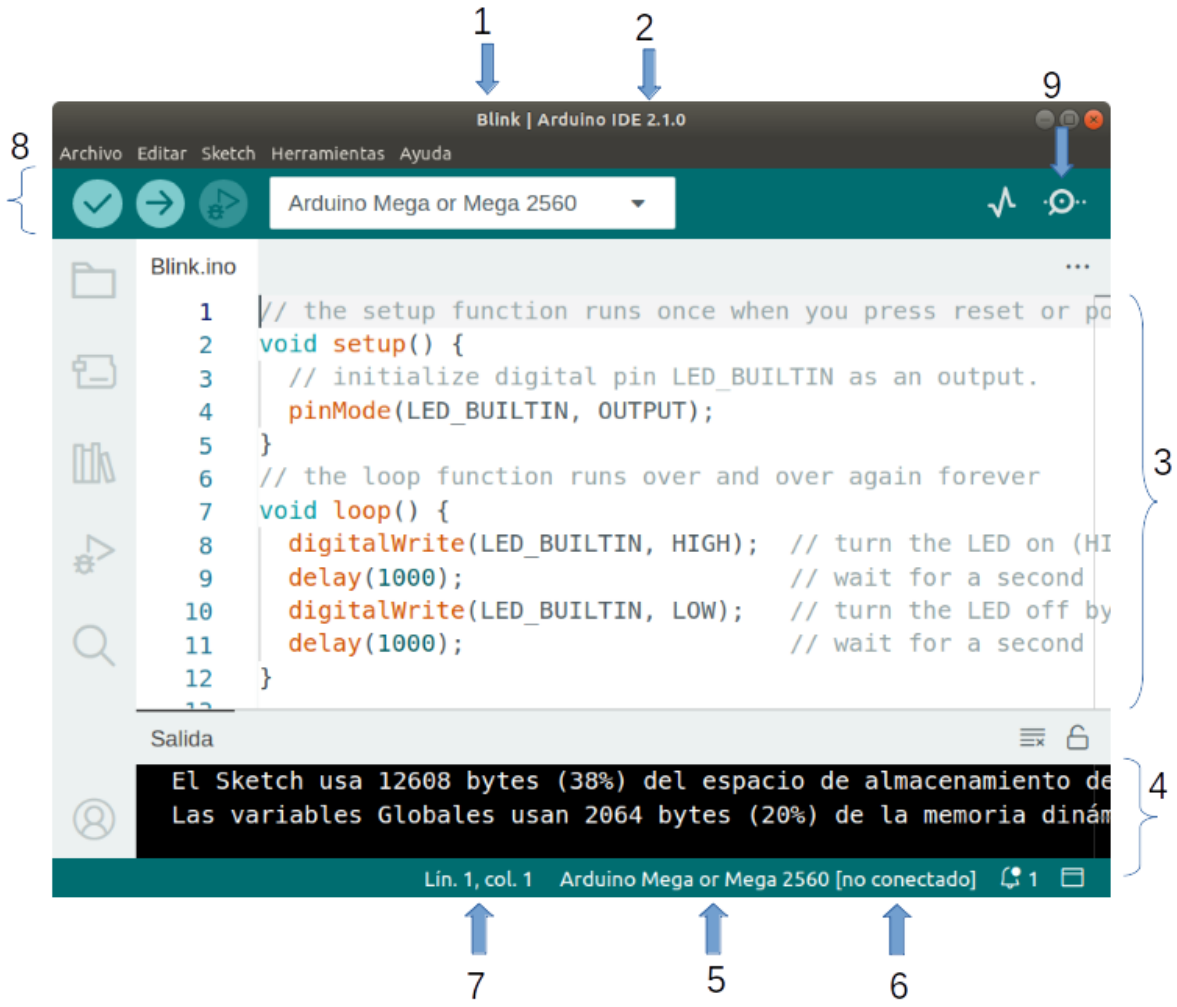
En esta página, encontrarás las opciones de descarga disponibles para diferentes sistemas operativos, como Windows, macOS y Linux. Selecciona la opción que corresponda a tu sistema operativo y haz clic en el enlace de descarga.

Una vez que se complete la descarga, sigue las instrucciones de instalación proporcionadas por Arduino para configurar el IDE en tu computadora. Después de la instalación, estarás listo para empezar a programar y cargar tus sketches en las placas Arduino.

Si surgen dudas con la instalación puedes visitar https://www.arduino.cc/en/Guide Install the Arduino Desktop IDE allí encontrarás una guía para cada sistema operativo.

Componentes del IDE

Sin importar el sistema sistema operativo que usemos el ide actual tiene estas características.



1 - Nombre del proyecto.
2 - Versión del IDE. (puede ser importante a la hora de solucionar problemas)
3 - Editor de código.
4 - Salida. Donde se reporta el estado de la compilación, errores, etc.
5 - Placa para la que se está compilando.
6 - Puerto de conexión.
7 - Posición del cursor.
8 - Verificación y Carga.
9 - monitor serie.

Variables y tipos de datos

Si conoces algún lenguaje como c/c++, Python o has usado Matlab, conocerás las variables.
Las variables son utilizadas para almacenar y manipular datos en la memoria (ram) durante la ejecución de un programa.

Antes de usar una variable debes declararla, indicando su tipo y nombre. Por ejemplo:

```
int cantidad;
float precio;
char letra;
```

Se declaran variables de tipo entero (int), punto flotante (float) y carácter (char), respectivamente.

Asignación de valores: Una vez declaradas, puedes asignar valores a las variables utilizando el operador de asignación (=). Por ejemplo:

```
cantidad = 25;
precio = 12.5;
letra = 'A';
```

Puedes utilizar las variables en expresiones y operaciones dentro del programa. Por ejemplo:

```
int suma = cantidad + 5;
float total = precio * suma;
```

En este caso, se realiza una operación de suma y multiplicación utilizando las variables cantidad y precio, respectivamente, y se almacena el resultado en las variables suma y total.
Se cuenta con varios tipos de datos para almacenar diferentes tipos de valores, como enteros, decimales, caracteres, booleanos, entre otros. Algunos tipos comunes son int (entero), float (punto flotante), char (carácter), bool (booleano), entre otros.
Es importante tener en cuenta el tipo de dato, también define el rango válido del contenido de la variable. El uso correcto de las variables es esencial para almacenar y manipular datos de manera eficiente y sin errores en tus programas.
Alcance de las variables: Las variables pueden tener un alcance local o global. Las variables locales se declaran dentro de una función y solo son visibles y accesibles dentro de esa función. Las variables globales se declaran fuera de cualquier función y son accesibles desde cualquier parte del programa.

Como se ve en los ejemplos cada línea termina con punto y coma, todas la variables están definidas con un tipo de dato. Éstas son reglas del lenguaje y se las denomina sintaxis.
La sintaxis es un conjunto de reglas y convenciones que definen la estructura y el formato válido de un lenguaje de programación. Estas reglas aseguran que el código sea comprensible para el compilador o intérprete correctamente.

Al iniciar en un lenguaje es necesario incorporar la sintaxis y para esto es útil tener a mano una “hoja de trucos”
Por ahora nos alcanza con este resumen:

Proyecto base

```
//descripción del proyecto
/*cómo funciona*/
//desarrollador
void setup() {
 //inicializaciones
}
void loop() {
 //código
}
```

Operadores

```
= ( asignación)
+ (adición) - (sustracción)
* (multiplicación)
/ (división) % (módulo)
== (igual a) != (distinto de)
< (menor que) > (mayor que)
<= (igual o menor que)
>= (mayor o igual que)
&& (y) || (o) ! (negación)
++ (incremento) -- (decremento)
+= (suma compuesta)
-= (resta compuesta)
```

Tipos de dato

```
void //vacío
boolean //(0, 1, true, false)
char //(ej. 'a' -128 a 127)
int //(-32768 a 32767)
long //(-2147483648 a 2147483647)
unsigned char //(0 a 255)
byte //(0 a 255)
unsigned int //(0 a 65535)
word //(0 a 65535)
unsigned long //(0 a 4294967295)
float //(-3.4028e+38 a 3.4028e+38)
double //(igual que los flotantes)
```

incluir librerías o archivos

```
#include <NombreDeLibreria.h>
#include “config.h”//ubicación
```

Definiciones

```
#define NOMBRE Valor
#define Pi 3.14
definiciones de Arduino.h
#define HIGH 0x1
#define LOW  0x0
#define INPUT 0x0
#define OUTPUT 0x1
#define INPUT_PULLUP 0x2
PI HALF_PI TWO_PI EULER SERIAL
```

Control de flujo

```
if(x < Pi) {
  //si x es menor que PII hacer }
else { // (opcional) }
while (x < 3) {
  //mientras x < 3 hacer }
for (int i = 0; i < 10; i++){
 //si i < a 10 hacer}
switch (var) {
case 1:
break;
case 2:
break;
default:
}
```

Funciones frecuentes

```
// I/O Digital
pinMode(pin,[INPUT, OUTPUT]);
digitalWrite(pin, valor);
int digitalread(pin);
//I/O Analógicas
int analogRead(pin);
analogWrite(pin, valor); //PWM
//puerto serie
Serie.begin(); int available();
byte read(); flush();
print(Datos);println(Datos);
//Números aleatorios
randomSeed(semilla); //long ó int
long random(max);
long random(min, max);
//Tiempo
delay(ms); unsigned long millis();
```

Se propone editar el ejemplo de Blink.

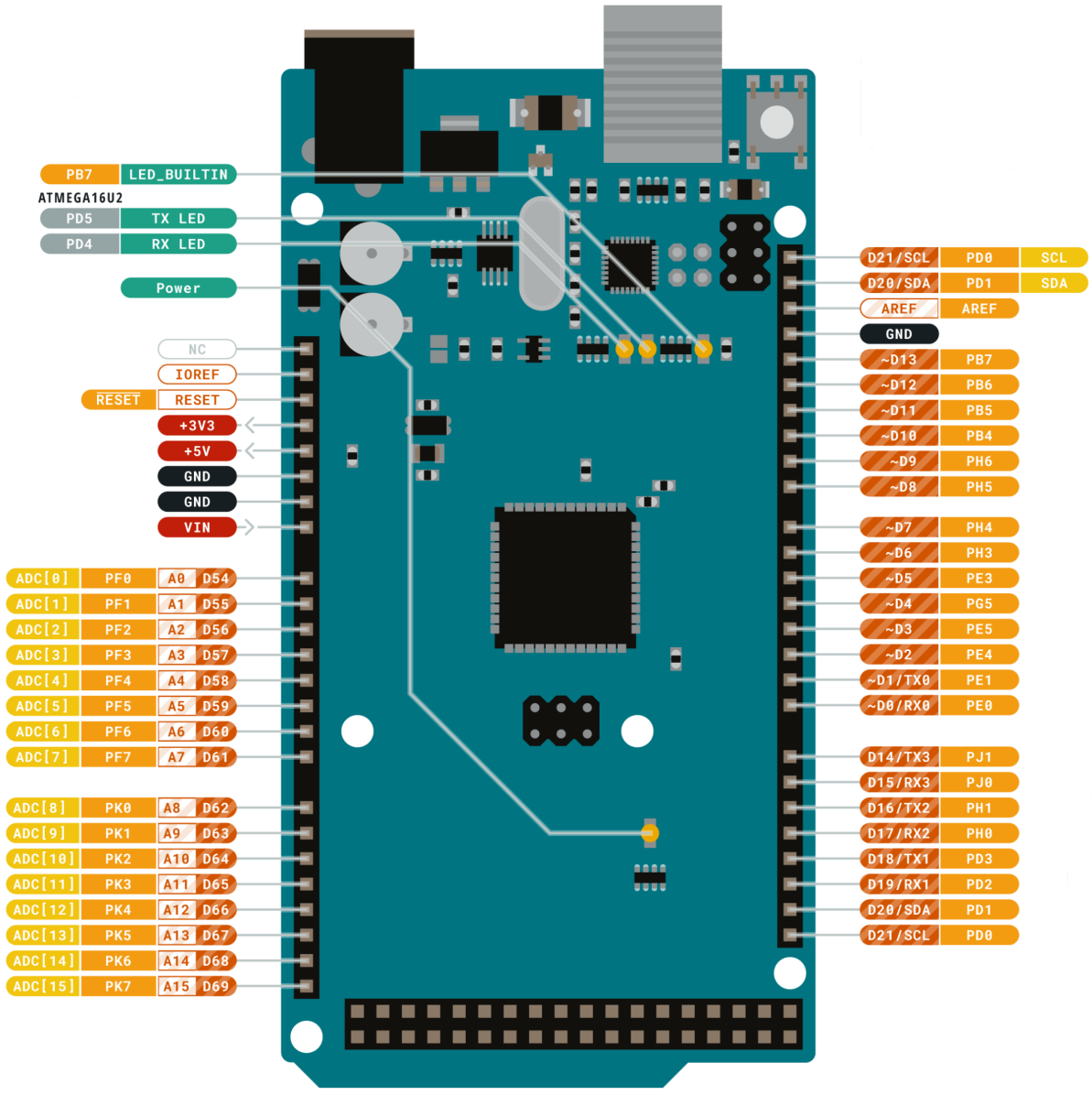
Modificar el programa para que parpadee con un periodo 1s 2s 3s 4s 5s y vuelva a iniciar.

```
#define PeriodoInicial 1000 //ms
```

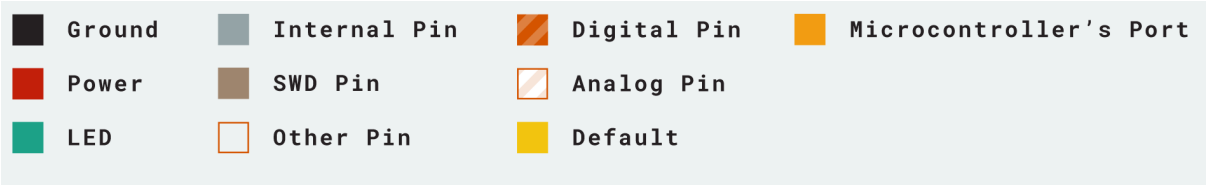
Esquemático - pinout

El Arduino Mega, la placa que disponemos, cuenta con un gran número de pines de entrada/salida, lo que te brinda una amplia flexibilidad para conectar componentes como LEDs, pulsadores, potenciómetros y LDRs. ¿Se podrían conectar en cualquier lugar?

Primero veamos un diagrama de la placa.



puede descargar el esquemático desde:
https://docs.arduino.cc/hardware/mega-2560
https://www.microchip.com/en-us/product/ATmega2560



En términos generales en muchas placas vamos a encontrar estas funcionalidades en los pines:

Alimentación:

VIN: Este pin permite alimentar la placa con un voltaje entre 7V y 12V a través de un adaptador externo. El voltaje se suministra a la placa a través del regulador de voltaje incorporado.
5V y 3.3V: Estos pines proporcionan voltajes de 5V y 3.3V, respectivamente. Se utilizan para alimentar circuitos integrados, sensores y otros componentes que funcionan a esos voltajes.
GND (Tierra): Estos pines se utilizan para completar los circuitos y proporcionar una referencia de voltaje común.

Comunicación:

Serial: Los pines RX0, TX0, RX1 y TX1 se utilizan para la comunicación serial UART (Universal Asynchronous Receiver-Transmitter) para enviar y recibir datos también para subir el código a la placa.
I2C: Los pines SDA y SCL se utilizan para la comunicación por bus I2C (Inter-Integrated Circuit) con dispositivos externos.
SPI: Los pines MISO, MOSI, SCK y SS se utilizan para la comunicación en serie síncrona (SPI) con dispositivos como pantallas LCD, tarjetas SD, entre otros.

Pines digitales:

Los pines digitales pueden funcionar tanto como entradas como salidas.
Se utilizan para conectar y controlar componentes digitales como LEDs, pulsadores, sensores digitales, relés, etc.
Pueden proporcionar un nivel de voltaje de 5V o 3.3V (HIGH) y 0V (LOW) en modo de salida.
Hay un total de 54 pines digitales en Arduino Mega (de 0 a 53) y funcionan de 0 a 5 V

Pines analógicos:

Los pines analógicos se utilizan para medir valores de voltaje analógico.
Con algún método comparar el voltaje de una señal y asociarlas con un valor representativo, que podamos manejar, proveniente de sensores como potenciómetros, LDRs, termistores, entre otros.
Arduino Mega cuenta con 16 pines analógicos (de A0 a A15) En este caso proporcionan una resolución de 10 bits, lo que significa que pueden leer valores entre 0 y 1023.
Estos pines también pueden funcionar como pines digitales si es necesario.
¿Se pueden usar como salidas analógicas?

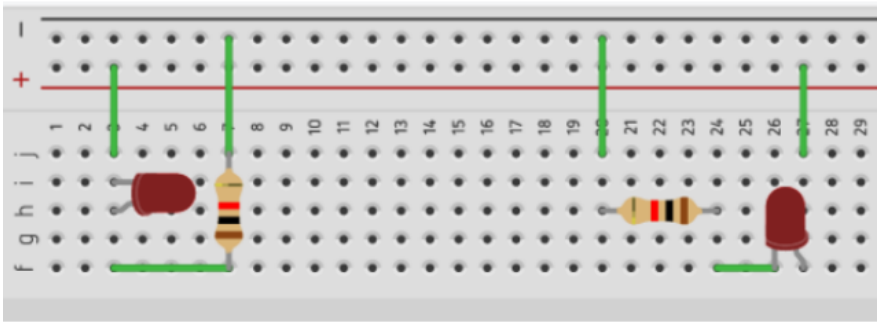
En documentación podemos encontrar que de un pin se pueden usar hasta 20mA de corriente. Ahora veremos cómo interpretar estos datos.

Protoboard.

Un Protoboard tiene una matiz de pequeños orificios. Estos agujeros permiten insertar fácilmente componentes electrónicos para hacer un prototipo (es decir, construir y probar una versión temprana de) un circuito electrónico.
Es una herramienta simple que se utiliza para conectar fácilmente los componentes eléctricos y los cables entre sí.
La distancia entre pines de muchos componentes es la que tiene el protoboard estándar pero para otros habrá que usar métodos diferentes al probar el circuito.

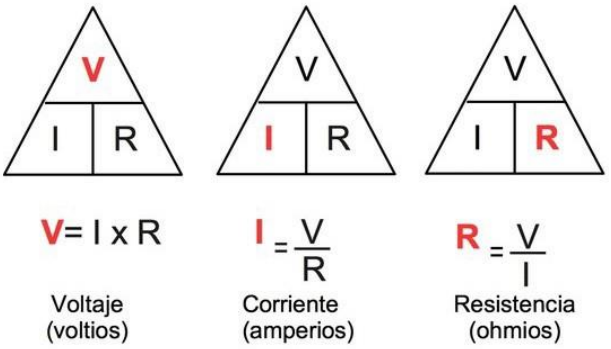
Es fácil imaginar la colocación de circuitos integrados, y otros componentes.
Las bandas laterales se usan como alimentación y pueden estar interrumpidas en la región media, permitiendo separar circuitos para alimentar con diferentes tensiones.
Los tramos cortos horizontales unen 5 agujeros, y solo estos, constituyendo un nodo.

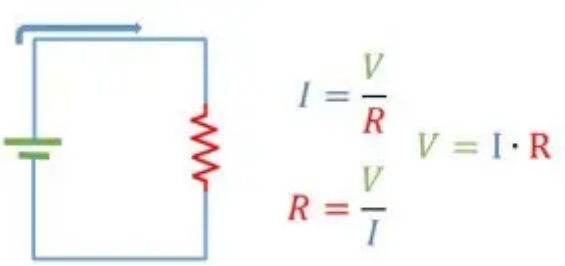
¿Cuál es la forma correcta? ¿Qué problema tiene la otra conexión?



Ley de Ohm

La ley de Ohm, postulada por el físico y matemático alemán Georg Simon Ohm, es una ley de la electricidad. Establece que la intensidad de la corriente I que circula por un conductor es proporcional a la diferencia de potencial V que aparece entre los extremos del citado conductor. Ohm completó la ley introduciendo la noción de resistencia eléctrica R; este es el coeficiente de proporcionalidad que aparece en la relación entre I y V.





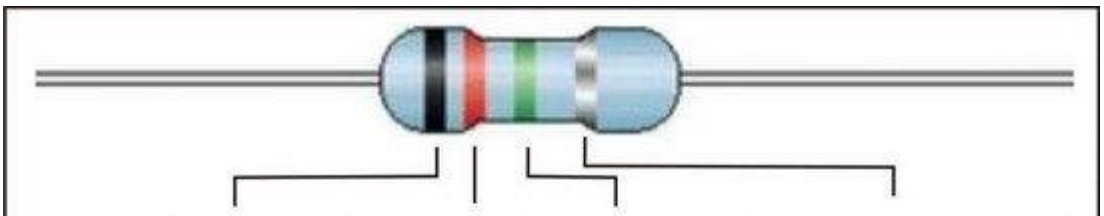
Resistencia

Se le denomina resistencia eléctrica a la igualdad de oposición que tienen los electrones al desplazarse a través de un conductor. La unidad de resistencia en el Sistema Internacional es el ohmio, que se representa con la letra griega omega

No es posible fabricar infinitos valores. por lo que comercialmente se dispone de los siguientes:

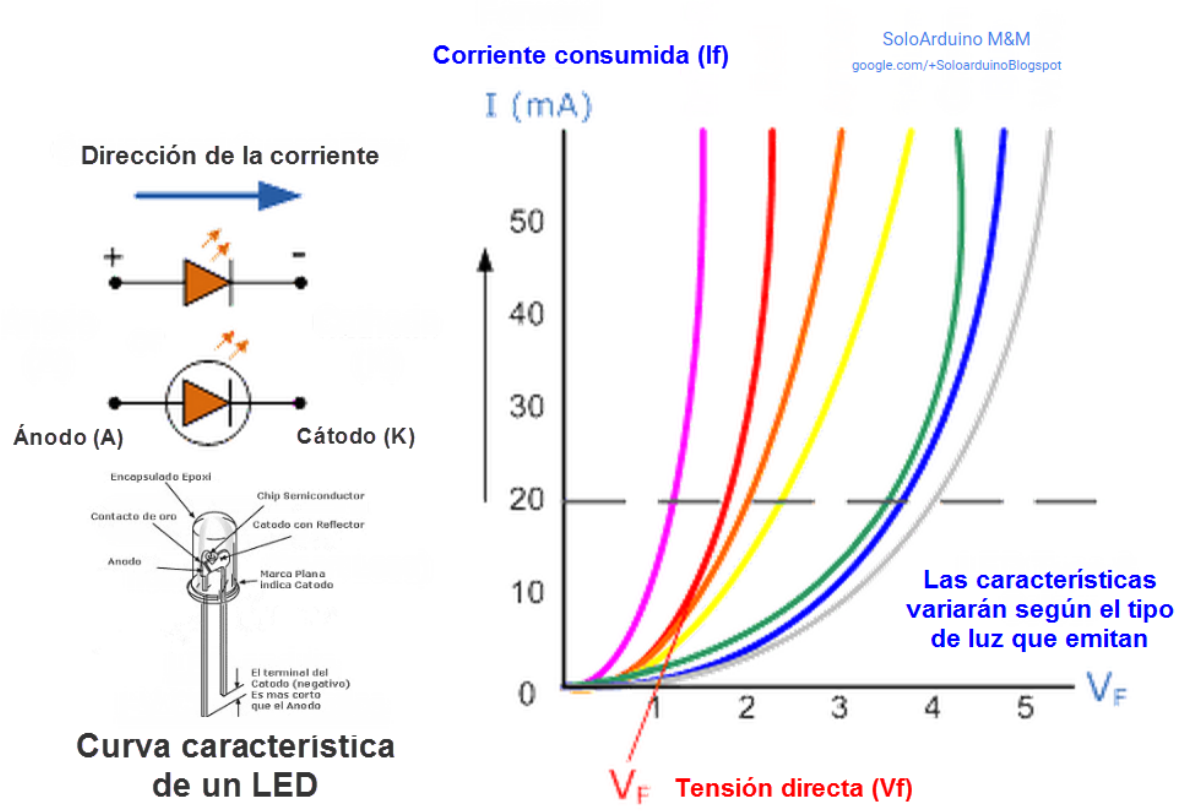
x 1	x 10	x 100	x 1.000 (K)	x 10.000 (10K)	x 100.000 (100K)	x 1.000.000 (M)
1 Ω	10 Ω	100 Ω	1 KΩ	10 KΩ	100 KΩ	1 M Ω
1,2 Ω	12 Ω	120 Ω	1K2 Ω	12 KΩ	120 KΩ	1M2 Ω
1,5 Ω	15 Ω	150 Ω	1K5 Ω	15 KΩ	150 KΩ	1M5 Ω
1,8 Ω	18 Ω	180 Ω	1K8 Ω	18 KΩ	180 KΩ	1M8 Ω
2,2 Ω	22 Ω	220 Ω	2K2 Ω	22 KΩ	220 KΩ	2M2 Ω
2,7 Ω	27 Ω	270 Ω	2K7 Ω	27 KΩ	270 KΩ	2M7 Ω
3,3 Ω	33 Ω	330 Ω	3K3 Ω	33 KΩ	330 KΩ	3M3 Ω
3,9 Ω	39 Ω	390 Ω	3K9 Ω	39 KΩ	390 KΩ	3M9 Ω
4,7 Ω	47 Ω	470 Ω	4K7 Ω	47 KΩ	470 KΩ	4M7 Ω
5,1 Ω	51 Ω	510 Ω	5K1 Ω	51 KΩ	510 KΩ	5M1 Ω
5,6 Ω	56 Ω	560 Ω	5K6 Ω	56 KΩ	560 KΩ	5M6 Ω
6,8 Ω	68 Ω	680 Ω	6K8 Ω	68 KΩ	680 KΩ	6M8 Ω
8,2 Ω	82 Ω	820 Ω	8K2 Ω	82 KΩ	820 KΩ	8M2 Ω
						10M Ω

físicamente las podemos identificar por las bandas de colores.
por ejemplo una resistencia con las bandas Amarillo Violeta Rojo = 47*100= 4700Ω o 4k7Ω
Código de colores de resistencias



Color	1ra. Banda	2da. Banda	3ra. Banda Multiplicador	Tolerancia %
Negro	0	0	x1	
Cafe	1	1	x10	
Rojo	2	2	x100	2%
Naranja	3	3	x1000	
Amarillo	4	4	x10000	
Verde	5	5	x100000	
Azul	6	6	x1000000	
Violeta	7	7	x10000000	
Gris	8	8	x100000000	
Blanco	9	9	x1000000000	
				Dorado 5%
				Plata 10%

led



Un LED (Light-Emitting Diode) es un componente electrónico que emite luz cuando se aplica una corriente eléctrica. Es una forma de diodo semiconductor que convierte la energía eléctrica en energía luminosa.
Son muy populares debido a su eficiencia energética, tamaño compacto, durabilidad y versatilidad. Se utilizan en una amplia gama de aplicaciones, desde indicadores luminosos en dispositivos electrónicos hasta iluminación de alta potencia en pantallas, lámparas, automóviles, señalización y más.

Están compuestos por una unión de materiales semiconductores de tipo P (con exceso de huecos o falta de electrones) y N (con exceso de electrones). Cuando se aplica una corriente eléctrica en la dirección correcta a través del LED, los electrones y huecos se recombinan en la unión P-N, liberando energía en forma de fotones (partículas de luz).

Se encuentran en diferentes colores, como rojo, verde, azul, amarillo y blanco, dependiendo de los materiales utilizados en su fabricación. También existen LEDs RGB (rojo, verde, azul) que pueden producir una amplia gama de colores mediante la combinación de diferentes intensidades de luz de cada color.

Se caracterizan por tener una vida útil mucho más larga que las bombillas incandescentes y las lámparas fluorescentes, además de ser más eficientes en cuanto al consumo de energía. Debido a sus ventajas, los LEDs han revolucionado la industria de la iluminación y se utilizan en una variedad de aplicaciones tanto en entornos domésticos como comerciales.

Es importante tener en cuenta las especificaciones del LED, como la tensión de encendido (forward voltage) y la corriente nominal (forward current), para seleccionar una fuente de alimentación adecuada. Estas especificaciones suelen indicarse en la hoja de datos del LED o en su embalaje.

Es necesario utilizar resistencias limitadoras de corriente en serie para asegurarte de que no se exceda la corriente nominal del LED. Las resistencias limitadoras se utilizan para ajustar la corriente que pasa a través del LED y protegerlo de daños.

color	voltaje
Rojo	1.6 a 2.2 V
Amarillo	1.6 a 2.2 V
Naranja	2.4 a 2.6 V
Azul	2.4 a 3.7 V
Verde	2.4 a 3.2 V
violeta	2.7 a 4.4 V
Blanco	2.8 a 3.5 V

Para calcular la corriente del LED, es necesario conocer la tensión de encendido (forward voltage, Vf) y la corriente nominal (forward current, If) del LED. Estos valores se pueden encontrar en la hoja de datos del LED o en las especificaciones proporcionadas por el fabricante.

Una vez que tienes estos valores, puedes utilizar la Ley de Ohm (V = I * R)
por lo que despejando R la fórmula para calcular la resistencia es:

R = (Vfuente - Vf) / If

Donde:

- Vfuente es la tensión de la fuente de alimentación (en voltios) que vas a utilizar para alimentar el LED.
- Vf es la tensión de encendido del LED (en voltios).
- If es la corriente nominal del LED (en amperios).
- R es la resistencia limitadora de corriente (en ohmios).

Una vez que hayas calculado el valor, debes seleccionar la resistencia comercial más cercana que sea igual o mayor que el valor calculado. Es importante seleccionar una resistencia con una potencia nominal adecuada para manejar la corriente del LED sin sobrecalentarse.
potencia: P = I * V

Recuerda que este cálculo es solo para una configuración simple de LED con una fuente de alimentación y una resistencia limitadora. Si estás utilizando múltiples LEDs en serie o paralelo, o si estás utilizando una fuente de alimentación regulada, pueden aplicarse cálculos o consideraciones adicionales.

Existen variedad de simuladores. En el siguiente enlace encontrarás uno que te permite usar un arduino, sensores y todo lo puedes conectar en un protoboard, subir tus códigos y probar tus proyectos.
https://www.tinkercad.com