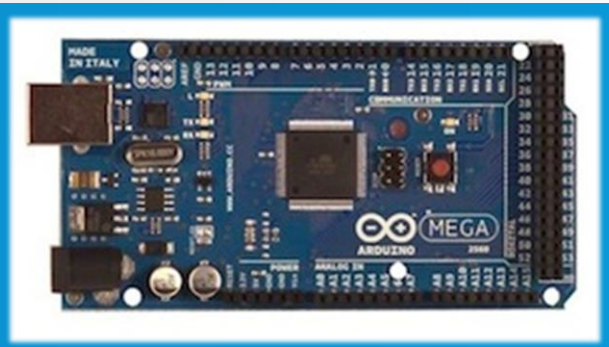
ARDUINO

- Placa de desarrollo y prototipado

  - Fácil de usar
  - Muchísimas comunidades para compartir ideas
  - Muchísimo soporte, ayuda y tutoriales disponibles
  - Placa es Open Hardware
  - Entorno de programación es OpenSource
  - Muchas variantes en constante aumento
  - ¡Limitado casi solamente por tu imaginación!

ALGUNAS VARIANTES

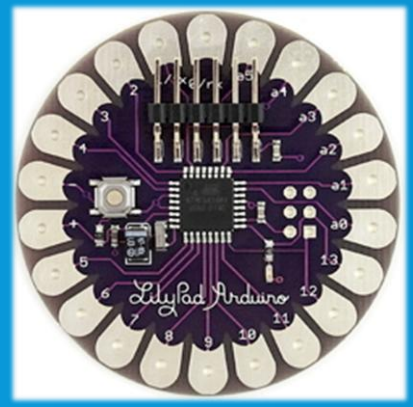


Arduino MEGA: Más prestaciones

Arduino ETHERNET: conexión cableada a internet

Arduino FIO: Comunicación por radiofrecuencia

Arduino UNO: Versátil, sencilla. La más usada para aprender



Arduino LILYPAD: Tecnología vestible

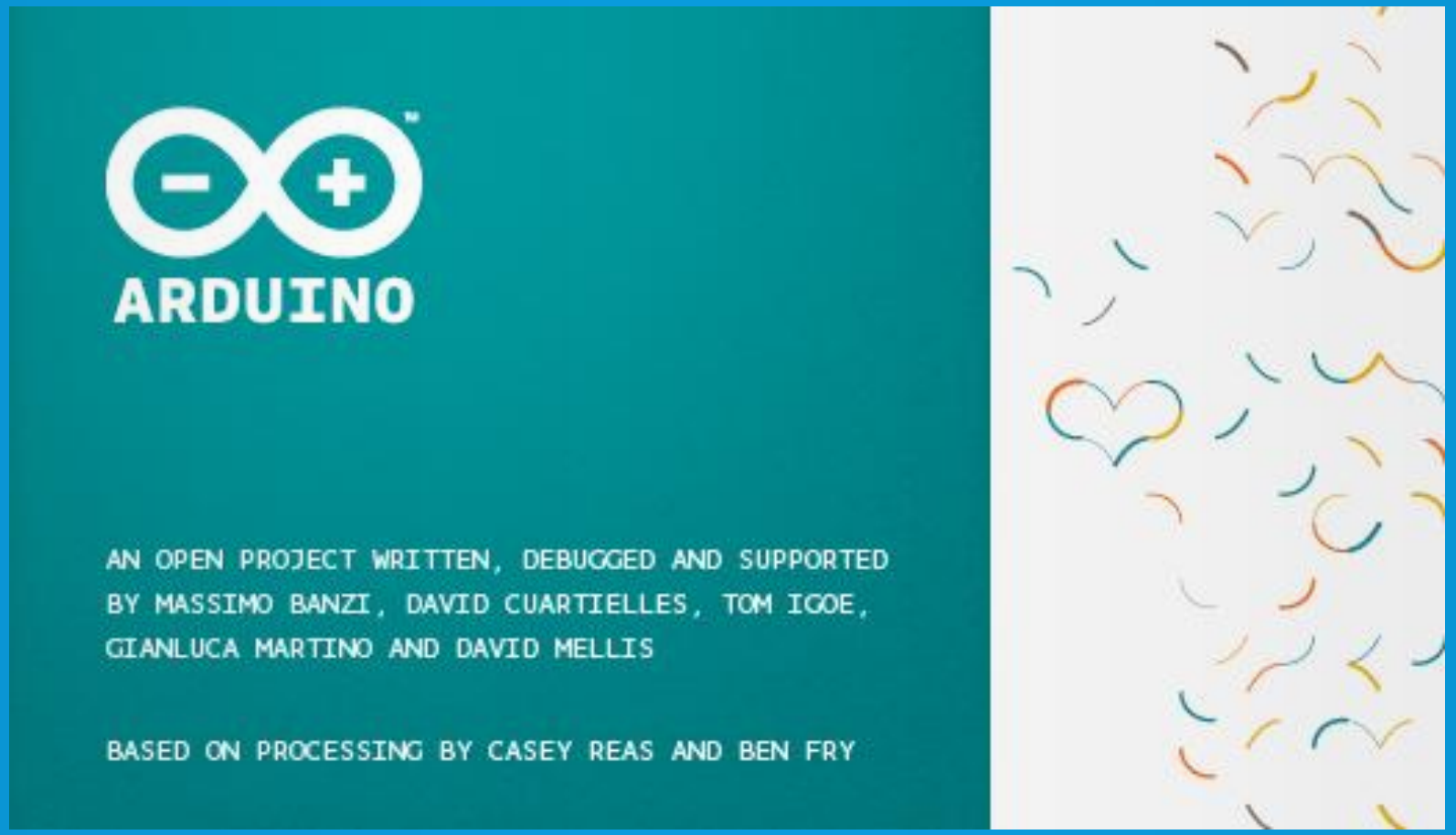
Arduino PRO MINI: Pequeñísima!

LA ESTRELLA: ARDUINO UNO

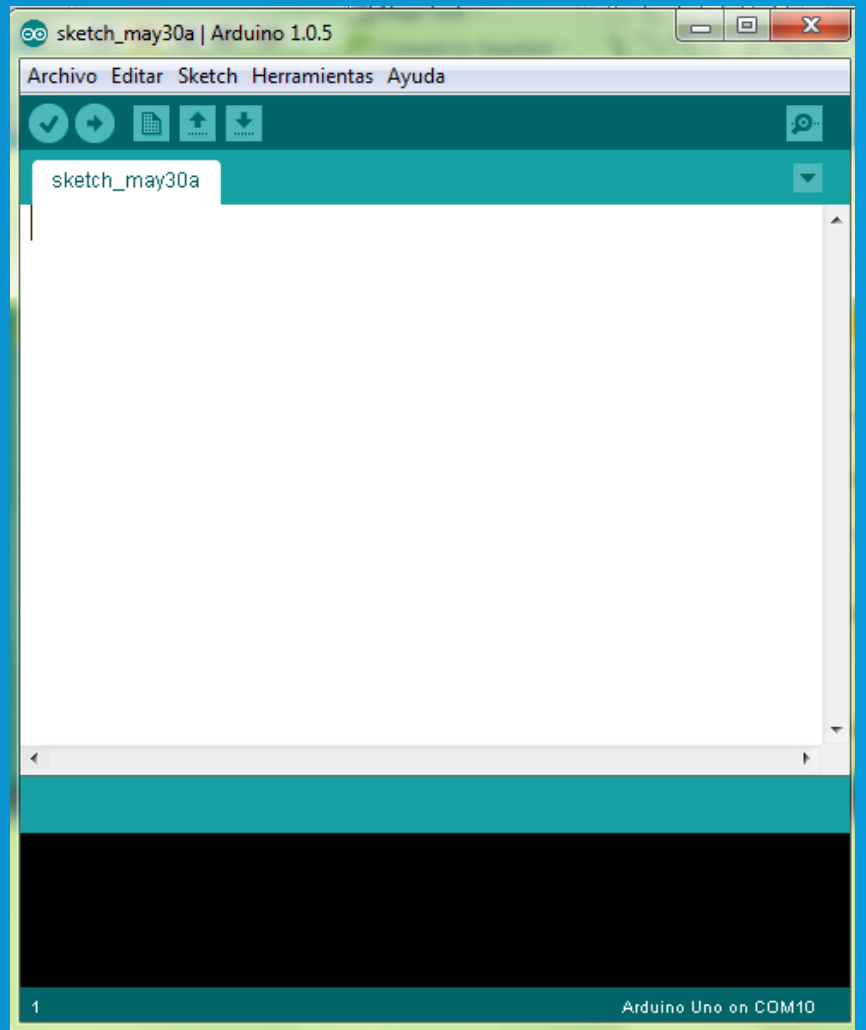
Resumen de características

Microcontrolador	ATMega328
Voltaje al que opera	5 V
Voltaje entrada (recomendado)	7-12 V
Voltaje entrada (límites)	6-20 V
Pines de Entrada/salida digitales	14 (de los cuales ,6 cuentan con PWM)
Pines de Entrada analógica	6
Corriente DC por pin de E/S	40 mA
Corriente DC para pin de 3.3 V	50 mA
Memoria Flash	32 KB (ATMega328)
SRAM	2 KB (ATMega328)
EEPROM	1 KB (ATMega328)
Velocidad del reloj	16 MHz

EL SOFTWARE: ARDUINO IDE



EL SOFTWARE: ARDUINO IDE



- Simple editor de texto
- Compila código c++ (lenguaje de programación) y lo carga al arduino
- Se tiene todo el poder de c++, además de las funciones propias de Arduino.
- ¡No es necesario saber c++! Basta con saber lo básico de programación.

¿CÓMO PROGRAMO EL ARDUINO?

- **Conceptos básicos de programación**
  - **Variables:** para guardar valores. Tienen un TIPO, nombre y un valor.

```
int mil=1000;    // variable de tipo int, nombre mil, valor 1000
char a= 'a';     // variable de tipo char, nombre a, valor 'a'
float pi=3.14;   // variable de tipo float, nombre pi, valor 3.14
int error=3.14; // No se puede hacer! Se asigna un float a un int
```

  - **Operaciones:** aritmética básica y operaciones lógicas entre variables y constantes

```
int dosmil=mil+mil;    // dosmil tiene valor 2000!
char b= 'a'+1;         // b vale 'b'. Los caracteres son números!
float pimedios=3.14/2;  // pimedios vale pi/2. Resultado float.
boolean mayor=(dosmil>mil); // mayor es 'true'
```

¿CÓMO PROGRAMO EL ARDUINO?

- **Funciones:** Encapsulan operaciones repetitivas. Reciben N parámetros con los que operar, y retornan un valor de cierto tipo, o también puede no retornar nada

```
int mult(int a,int b){
        return a*b;
}
//mult(a,b); retornará el valor de a*b

float globalnum=0;

void guardaNum(float num){
        globalnum=num;
}
//guardaNum(num) copia num en globalnum, y luego no retorna nada.
```

¿CÓMO PROGRAMO EL ARDUINO? (¡ÚLTIMA!)

- **IF – ELSE:** Dada cierta condición: si ésta se cumple, se ejecuta un código. Si no, se ejecuta otro código.

```
boolean se_cumple,no_se_cumple;
if (mil==1000){
        se_cumple=true;
        no_se_cumple=false;
}
else{
        se_cumple=false;
        no_se_cumple=true;
}
// Sólo una de estas dos hebras se ejecutará!
```

¿CÓMO PROGRAMO EL ARDUINO? (¡AHORA SI QUE LA ÚLTIMA!)

IMPORTANTE: Arduino exige iniciar todo programa con las siguientes funciones declaradas:

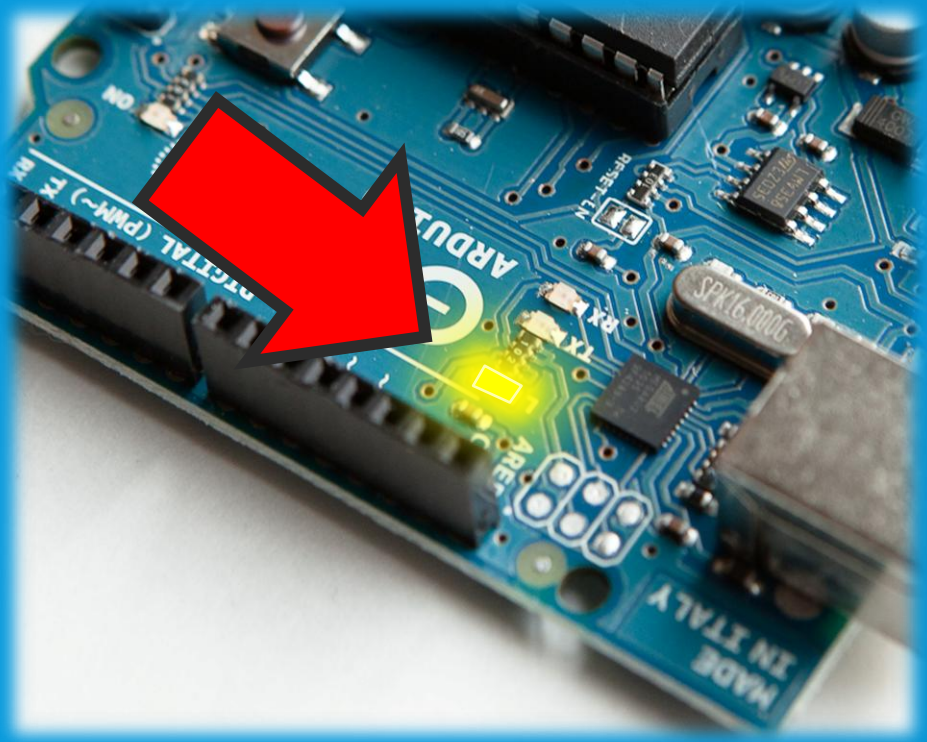
```
void setup(){
// Lo que se ponga acá, se ejecutará cuando el arduino inicie

}
void loop(){
// Lo que se ponga acá se ejecutará luego de haber ejecutado
//setup(), y se ejecutará infinitas veces hasta desconectarlo.
}
```

Sin estas funciones, el compilador se quejará.

EXPERIENCIA 1: "BLINK" (PARPADEO)

A poner las manos en la masa!

- **Meta:** Lograr que el LED integrado en arduino parpadee cada 1 segundo.
- **Materiales:**
  - 1 Arduino UNO
  - 1 Cable USB
- **Instrucciones:**
  ¡Pongan atención a continuación!



EXPERIENCIA 1: "BLINK" (PARPADEO)

Funciones útiles / necesarias para la experiencia:

- `pinMode(p, modo)`

Quieres LEER desde el pin `p`? Entonces `modo` = INPUT

Quieres ESCRIBIR hacia el pin `p`? Entonces `modo` = OUTPUT

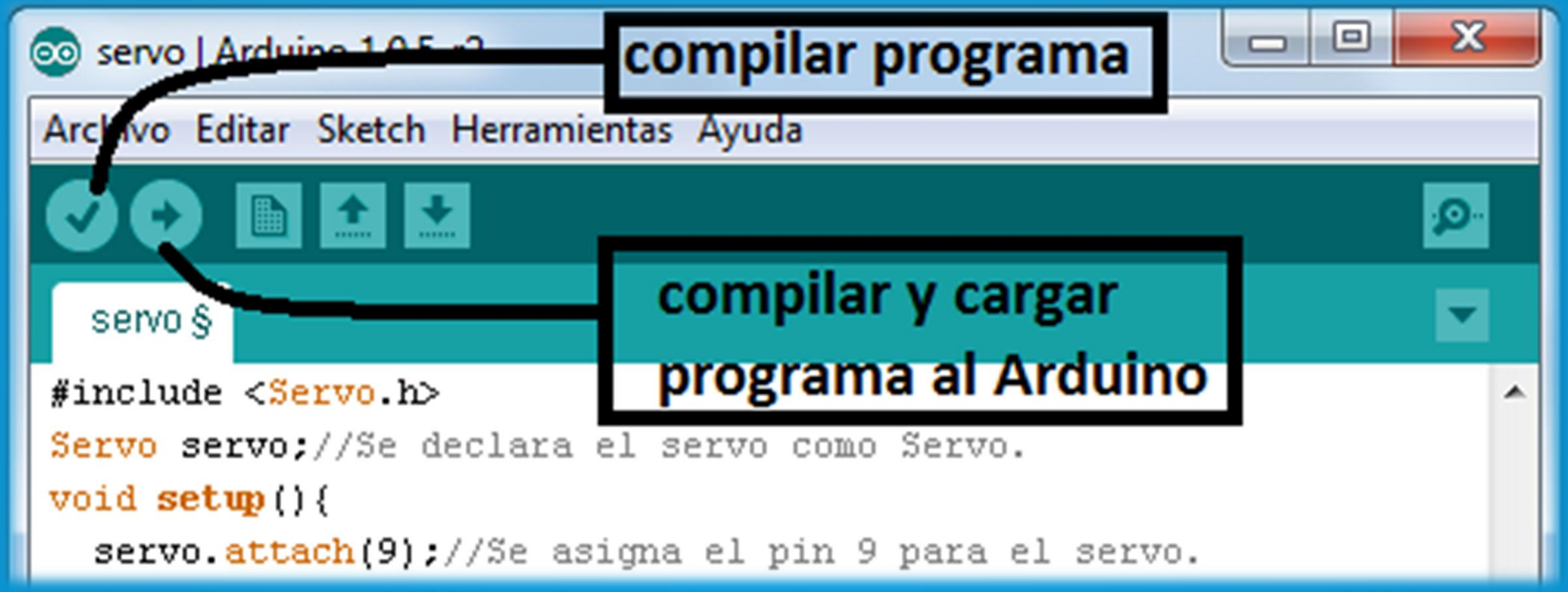
- `digitalWrite(p, v)`

Escribir valor `v` en pin `p`. `v` puede valer HIGH o LOW

- `delay(t)`

Arduino se "duerme" por `t` milisegundos

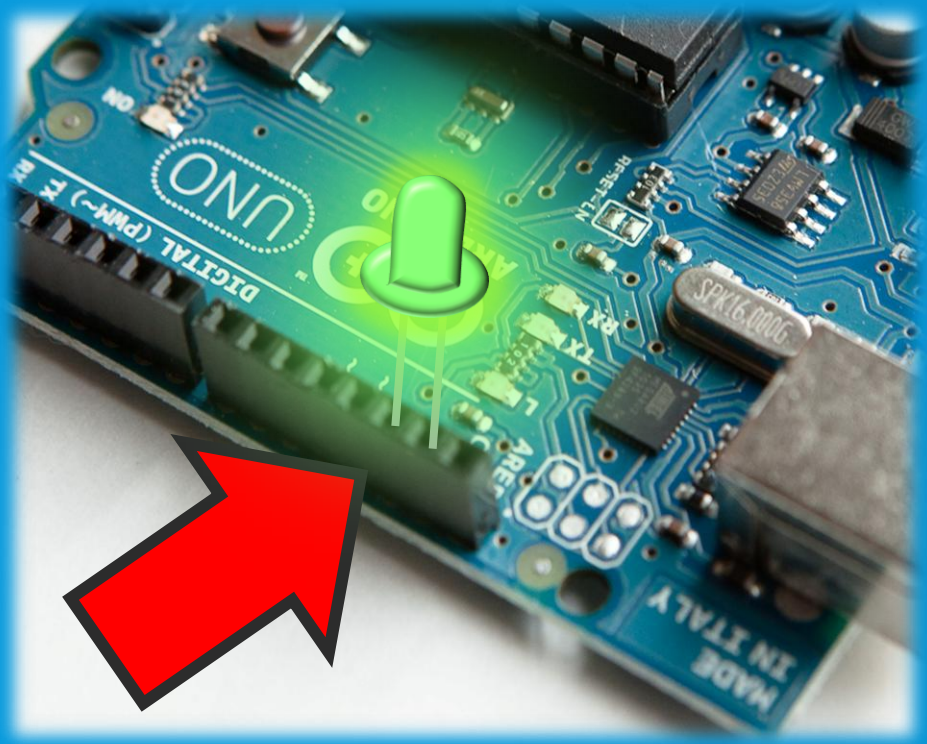
EXPERIENCIA 1: "BLINK" (PARPADEO)



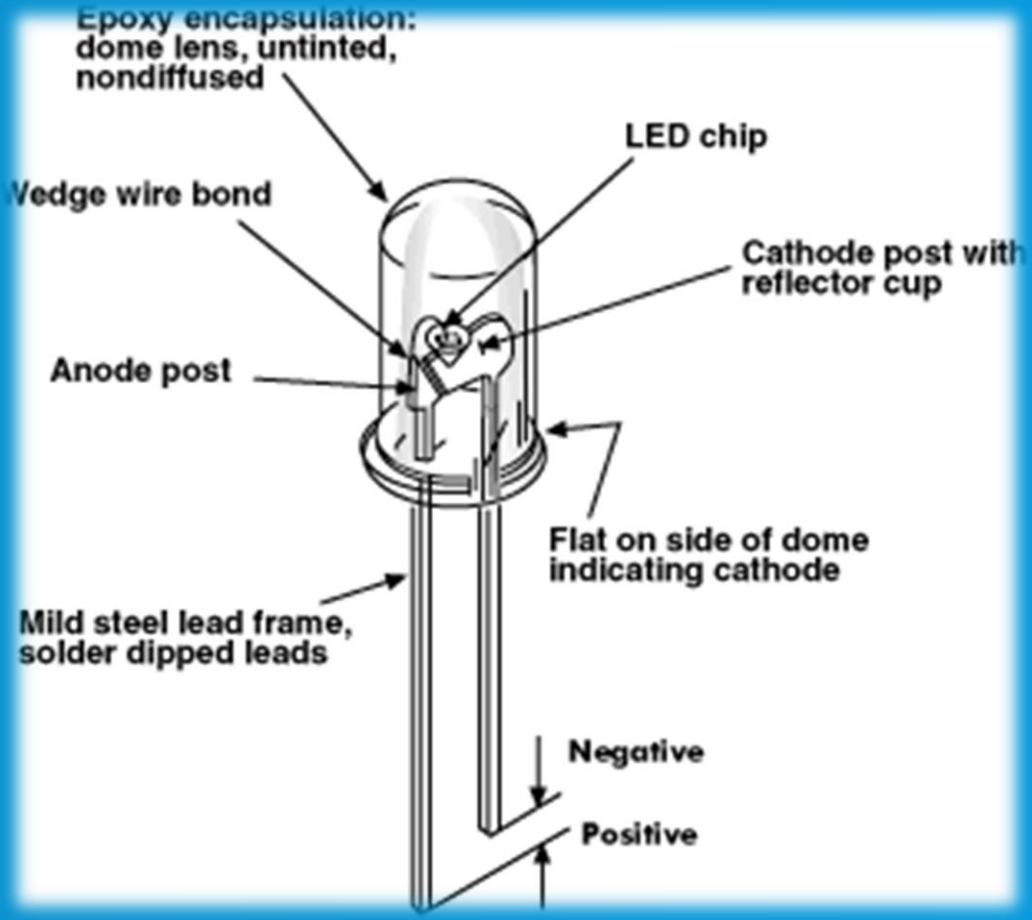
EXPERIENCIA 1.5: "BLINK" CON LED EXTERNO

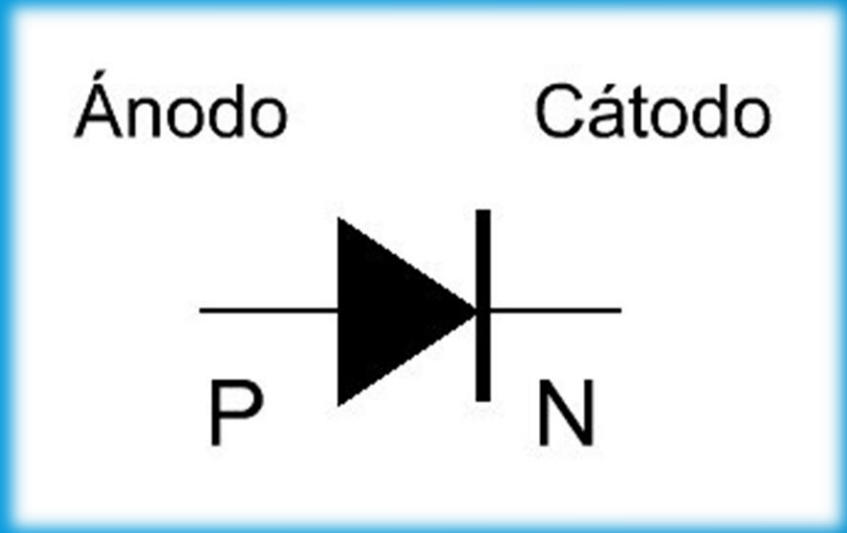
Funciona! Ahora usemos un LED externo.

- **Meta:** Lograr que el LED que tu elijas parpadee cada 1 segundo.
- **Materiales:**
  - 1 Arduino UNO
  - 1 Cable USB
  - 1 diodo LED
- **Instrucciones:**
  ¡Pongan atención a continuación!



EXPERIENCIA 1.5: "BLINK" CON LED EXTERNO





EXPERIENCIA 1.5: "BLINK" CON LED EXTERNO

ADVERTENCIA

Sólo usar pin 13 para esta experiencia.
Único pin con RESISTENCIA INTEGRADA.

A no ser que quieran freir sus pines y componentes…

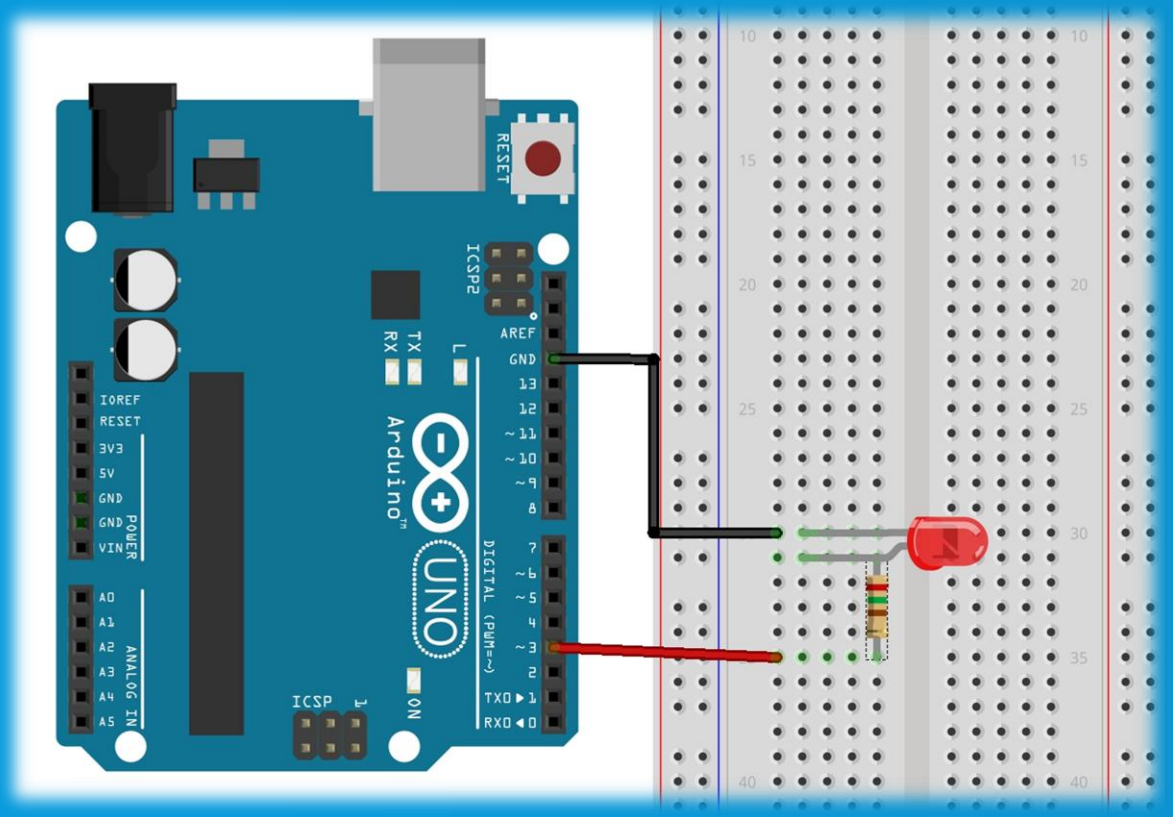
Explicación: Ley de OHM

$$IxR=V$$

EXPERIENCIA 2: "FADE"

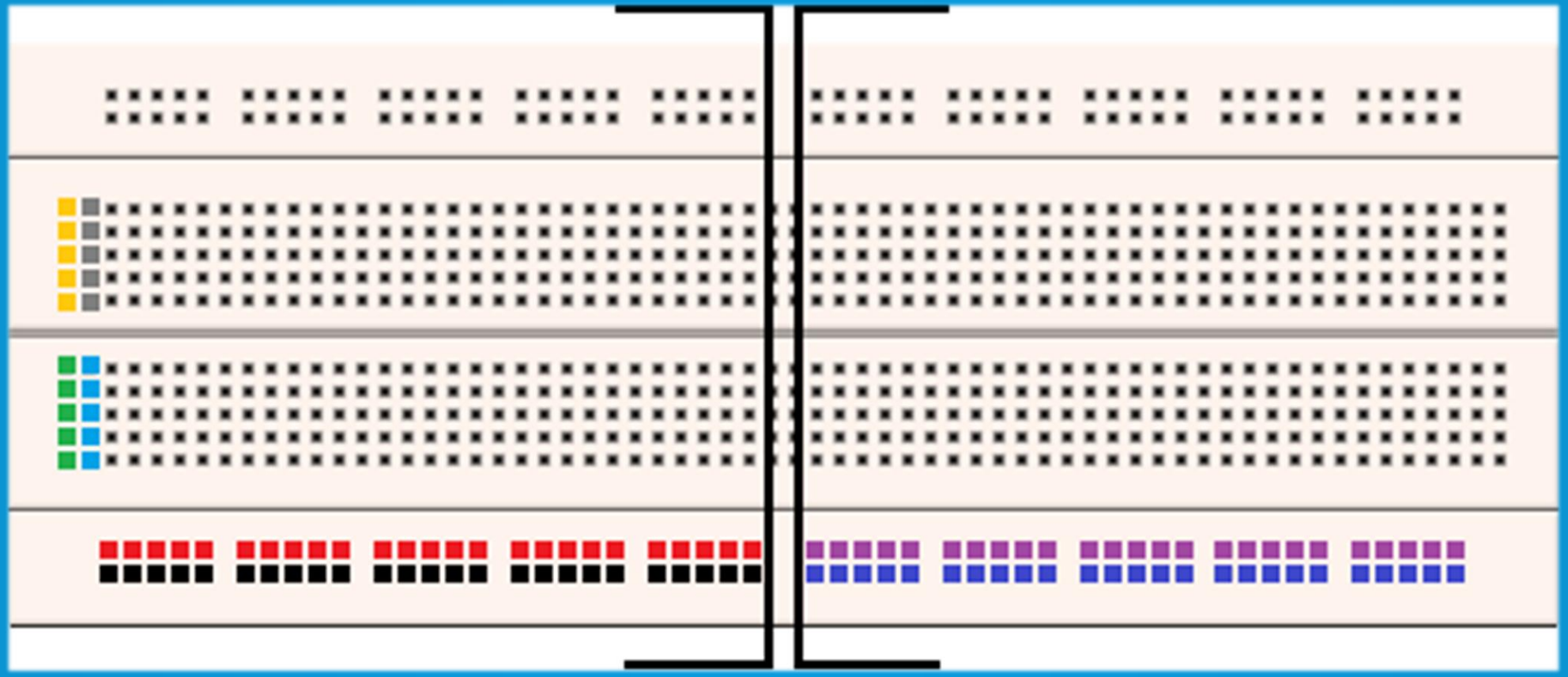
Y si quiero un led a brillo medio?

- **Meta:** Lograr que el LED se apague y prenda de forma suave.
- **Materiales:**
  - 1 Arduino UNO
  - 1 Cable USB
  - 1 protoboard
  - 1 diodo LED
  - 1 resistencia, de 250 Ω aprox.
  - Jumpers (cables)



EXPERIENCIA 2: "FADE"

Estructura de la protoboard



EXPERIENCIA 2: "FADE"

Funciones útiles / necesarias para la experiencia:

- `analogWrite(p, v)`

Escribir valor `v` en pin `p`. `v` es un número de 0 a 255.

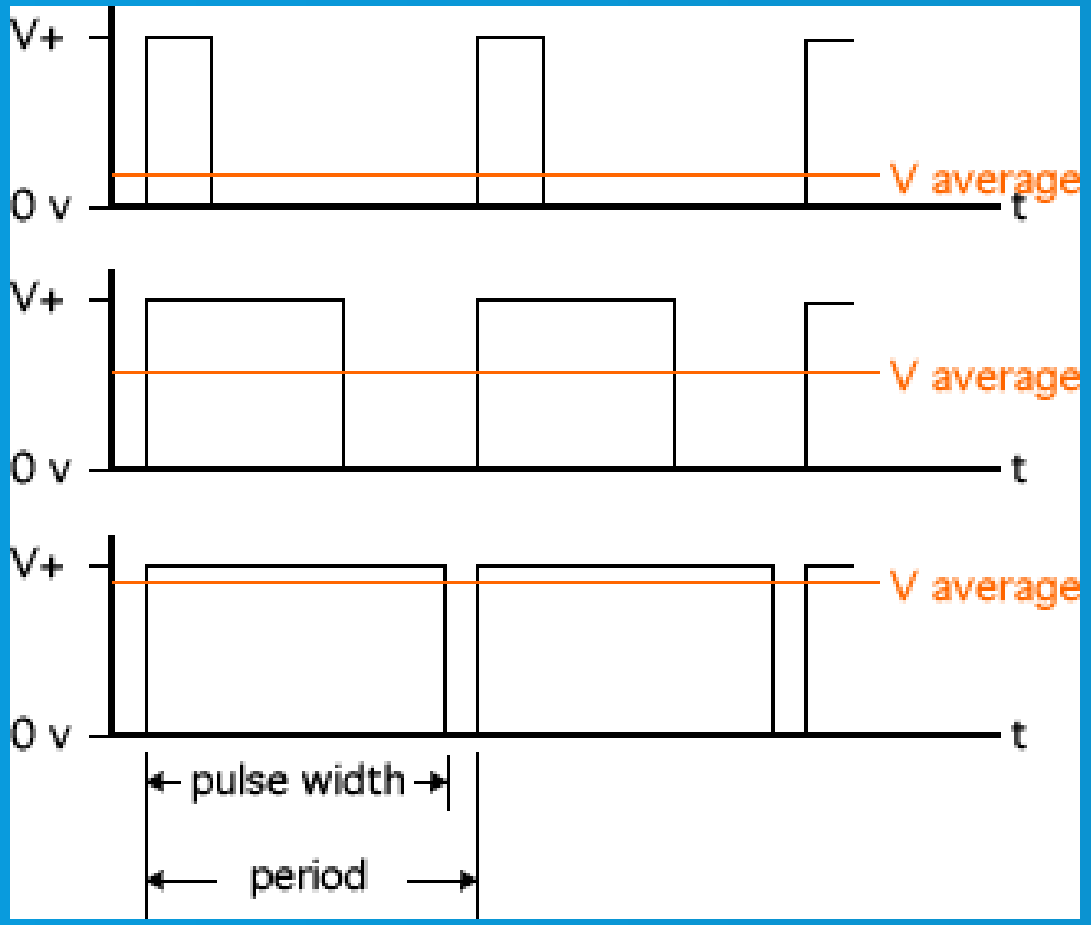
EXPERIENCIA 2: "FADE"

Explicación: PWM (Pulse Width Modulation)

Técnica utilizada por microcontroladores para emular voltajes intermedios a GND y VCC.

¡Arduino sólo sabe enviar 0 y 5 volts!

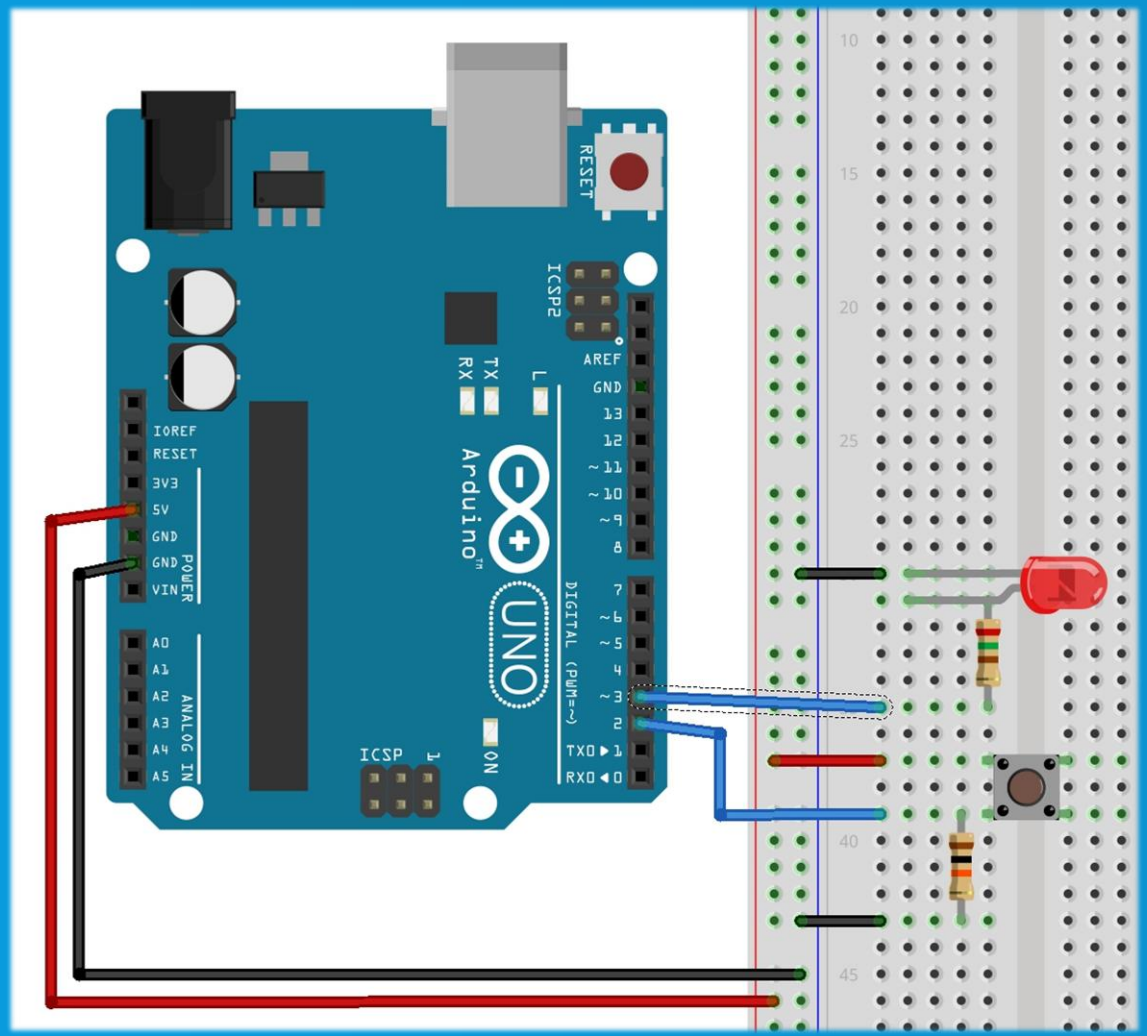
Ver pizarra.



EXPERIENCIA 3: "PULSADOR"

¿Y si queremos añadir un botón?

- **Meta:** Lograr que el LED se apague y prenda usando un botón
- **Materiales:**
  - 1 Arduino UNO
  - 1 Cable USB
  - 1 protoboard
  - 1 diodo LED
  - 2 resistencias, de 250 Ω y 10 KΩ aprox.
  - 1 botón
  - Jumpers (cables)



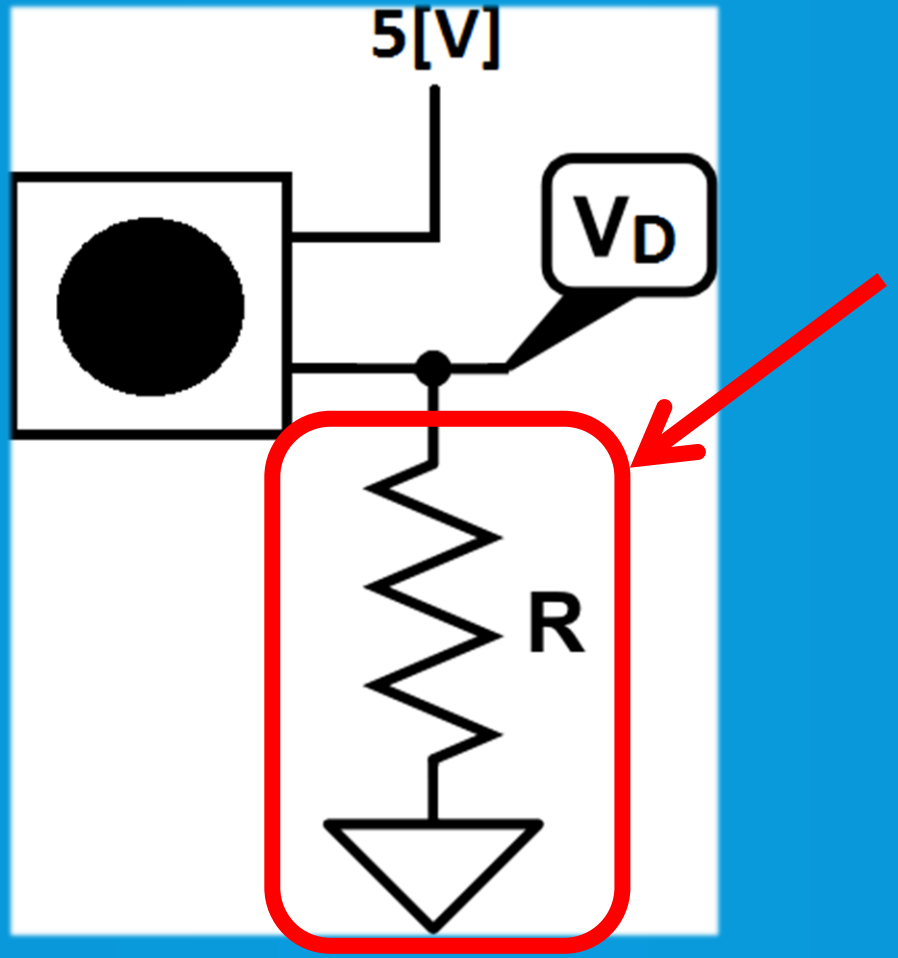
EXPERIENCIA 2: "FADE"

Funciones útiles / necesarias para la experiencia:

- `digitalRead(p)`

Leer voltaje en pin `p`. La función retorna HIGH si el voltaje en el pin es mayor que 2.5V y LOW en caso contrario.

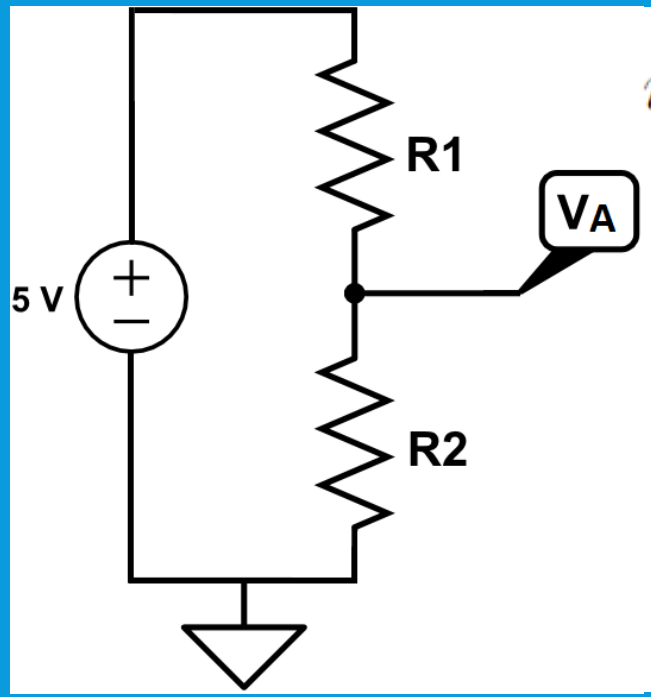
EXPERIENCIA 3: "PULSADOR"



¿Por qué es necesario?
Poner atención.

ANTES DE LA EXPERIENCIA 4…

Divisor de voltaje



$$i = \frac{5[V]}{R_1 + R_2} = \frac{V_A}{R2} \Rightarrow V_A = \frac{R_2}{R_1 + R_2} 5[V]$$

Este cálculo se obtiene de aplicar ley de OHM en ambas resistencias, y sabiendo que la corriente que circula por ambas es igual.

ANTES DE LA EXPERIENCIA 4…

Divisor de voltaje

- Valores para *R1* y *R2* definen *Va*…

Fijando una resistencia y variando la otra, voltaje *Va* varía.

- Arduino puede leer voltajes de 0 a 5 volts desde sus *pines analógicos.*
- Los interpreta como valores de 0 a 1023 respectivamente.

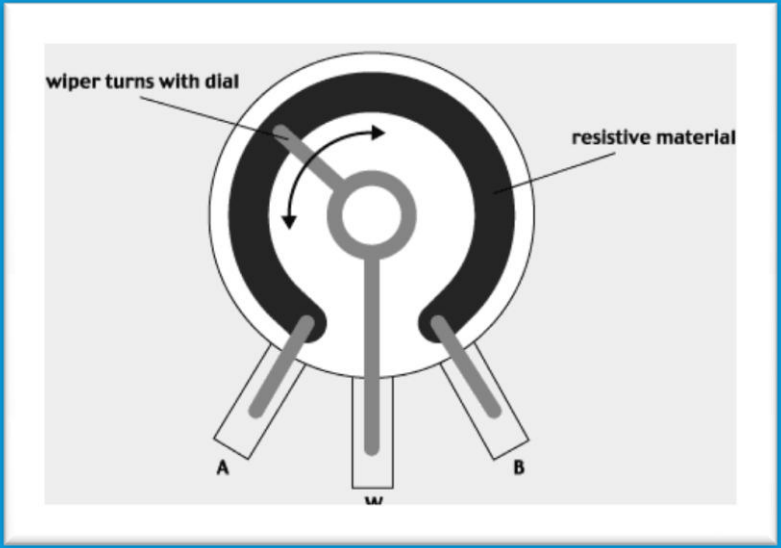
¿Cómo variar una resistencia?

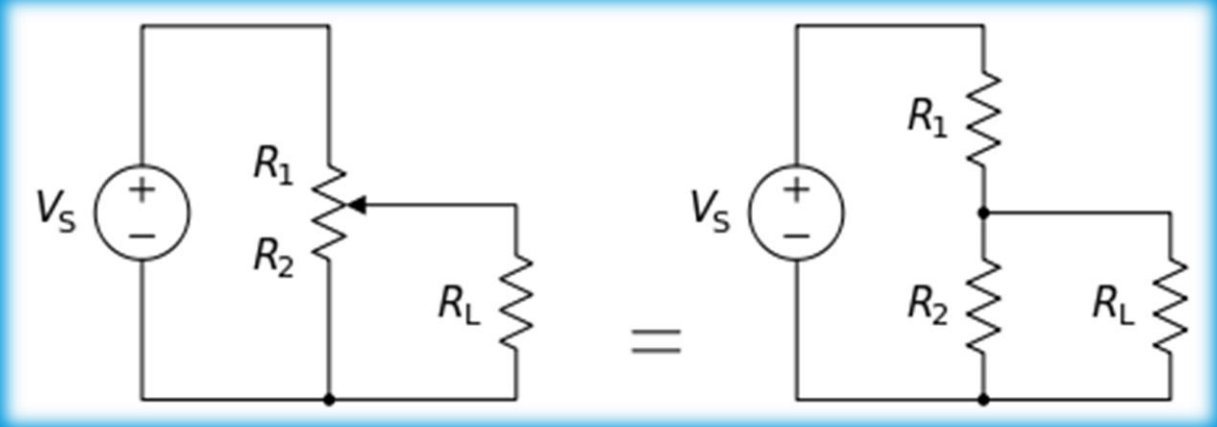
¡Hay muchos tipos de resistencias variables!

ANTES DE LA EXPERIENCIA 4…

Divisor de voltaje

Ejemplo: Potenciómetro

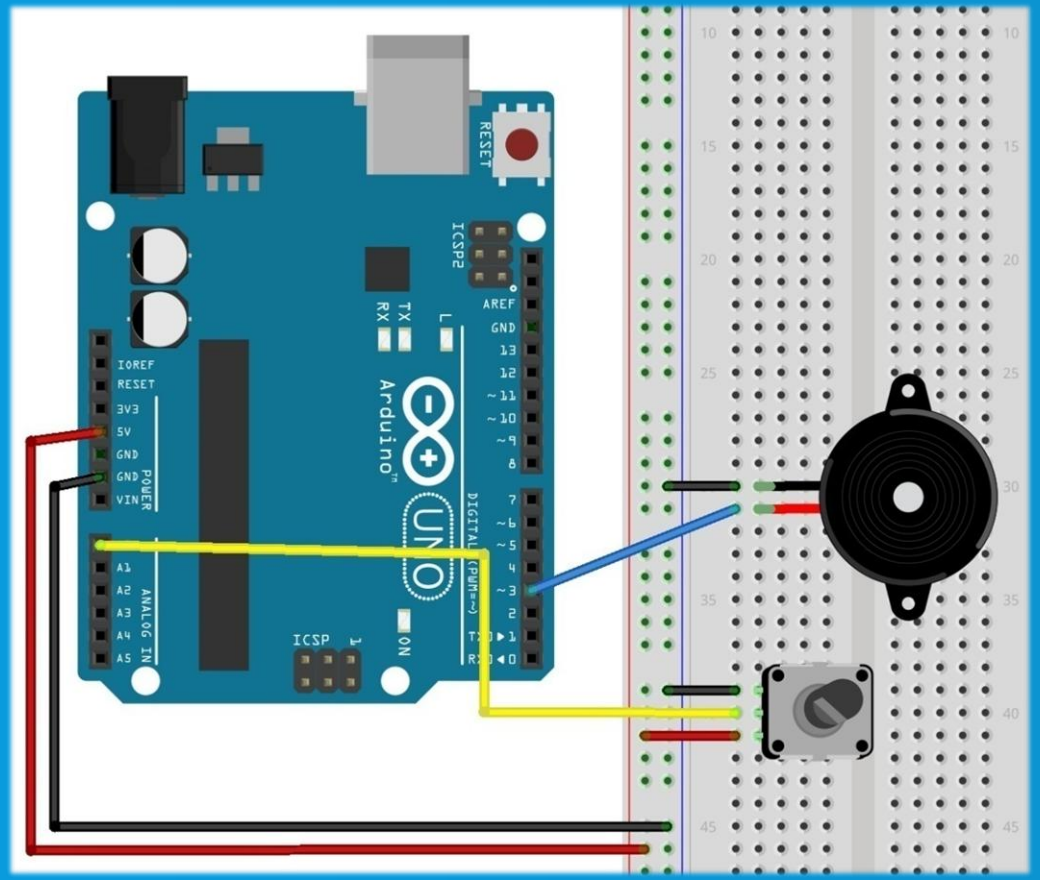




EXPERIENCIA 4: "LECTURA ANALÓGICA"

Leer valores de voltaje, y hacer algo con ellos

- **Meta:** Reproducir notas
- **Materiales:**
  - 1 Arduino UNO
  - 1 Cable USB
  - 1 protoboard
  - 1 diodo LED
  - 1 potenciómetro
  - 1 buzzer
  - Jumpers



EXPERIENCIA 4: "LECTURA ANALÓGICA"

Buzzer: Parlante sencillo.

¡Ojo con el signo!

Funciones útiles / necesarias para la experiencia:

- `analogRead(p)`

Leer voltaje del pin `p`. Retorna un número de 0 a 1023.
¡Sólo funciona con pines analógicos!

- `tone(p,freq)`

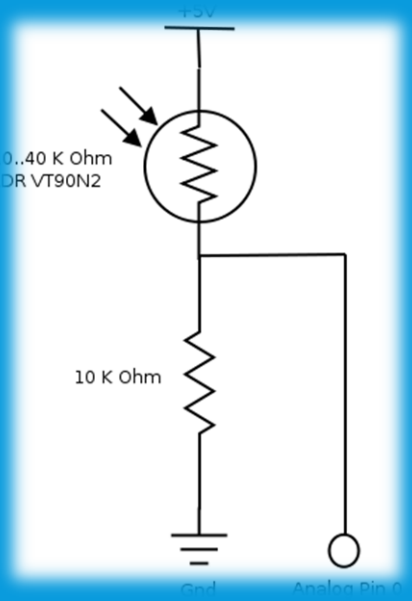
Hace sonar pin `p` con un sonido de frecuencia `freq`.

ANTES DE LA EXPERIENCIA 4…

Divisor de voltaje

Otro ejemplo de resistor variable: **LDR (Fotorresistencia)**

Varía su resistencia según luminosidad en el ambiente.



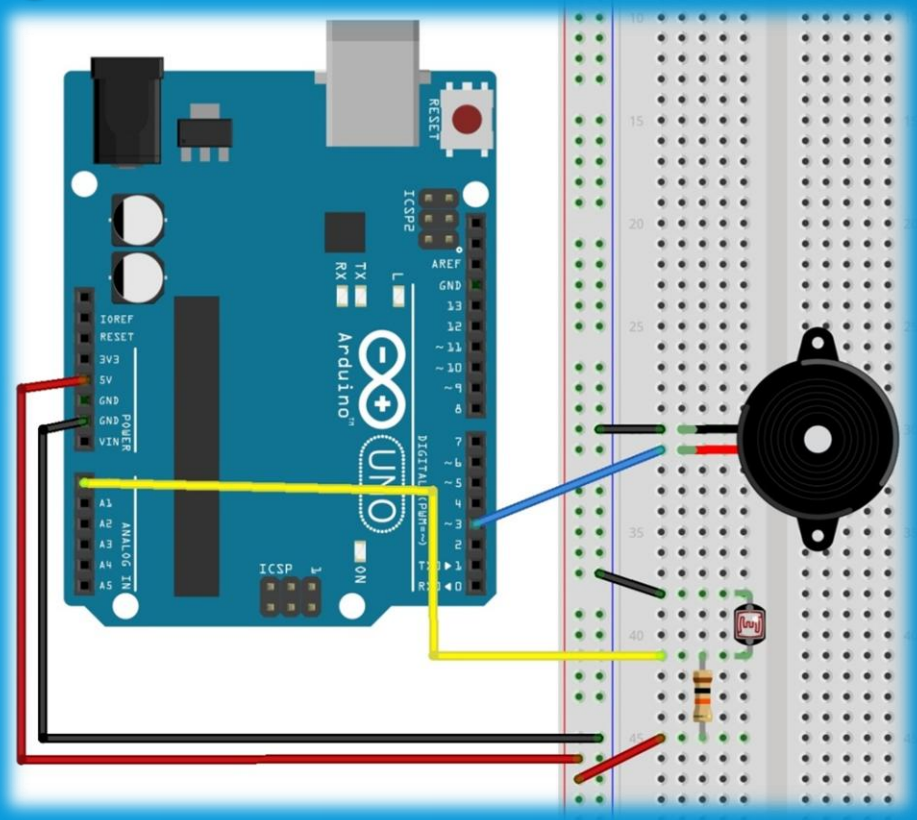
Aplicada a un divisor de voltaje…

¡Se puede fabricar un sensor de luz!

EXPERIENCIA 4.5: "SENSOR DE LUZ"

Leer valores de voltaje, y hacer algo con ellos

- **Meta:** Reproducir notas que varíen según luminosidad del ambiente.
- **Materiales:**
  - 1 Arduino UNO
  - 1 Cable USB
  - 1 protoboard
  - 1 diodo LED
  - 1 resistencia de 10 kΩ
  - 1 fotorresistor
  - 1 buzzer
  - Jumpers



EXPERIENCIA 4.5: "SENSOR DE LUZ"

Funciones útiles / necesarias para la experiencia:

- **`map(n,ai,af,bi,bf)`**

Lleva el valor **`n`**, que va desde **`ai`** a **`af`**, hacia valores que vayan de **`bi`** a **`bf`**.
Ejemplo: si **`n`** va de 0 a 1024 y quiero que vaya de 0 a 255, hago map(**`n`**,0,1024,0,255). **`bi`** y **`bf`** pueden ser un intervalo invertido también.
Ejemplo: map(**`n`**,0,1024,255,0).
Es una función meramente numérica.

- **`constrain(n,a,b)`**

Si **`n`** está en [**`a`**,**`b`**], retorna **`n`**. Sino, retorna **`a`** o **`b`** según dónde **`n`** se sale del intervalo.