



LABORATORIO 3 - DF - FCEyN - UBA  
1<sup>er</sup> cuatrimestre - 2023

# ¿Qué es Arduino?

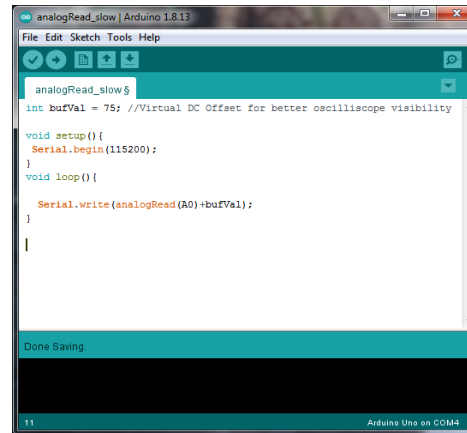
Plataforma de creación de electrónica

[www.arduino.cc](http://www.arduino.cc)

Placa Arduino



Software de programación  
(Arduino IDE)



## Hardware y software abiertos

El proyecto nació en 2003, impulsado por **estudiantes** del Instituto de Diseño Interactivo de Ivrea, **Italia**, con el fin de facilitar el acceso y uso de electrónica y programación.

¿Cómo funciona?

Entradas

→ Procesamiento

→ Salida

Analógicas (conversor ADC)  
Digitales (lógicas)

Microprocesador  
Comunicación con PC

Interruptores  
Control de motores  
Señales

...

# Arduino UNO: conversor ADC

Precio Aproximado \$ 4000



- Conversor Analógico Digital (ADC) 10 bits
- 6 Entradas analógicas Rango [0 ,5] V
- Sample rate aprox. 10 kS/s (0.1 ms de tiempo de medición)

10 bit  $\rightarrow$   $2^{10} = 1024$  divisiones

`analogRead()` devuelve un número entre 0 y 1023

Donde 0 es 0 V y 1023 es Vref

- Se puede usar un voltaje de referencia distinto al 5 V por defecto en el pin Vref
- Con Vref = 5 V, la sensibilidad es de 4.9 mV

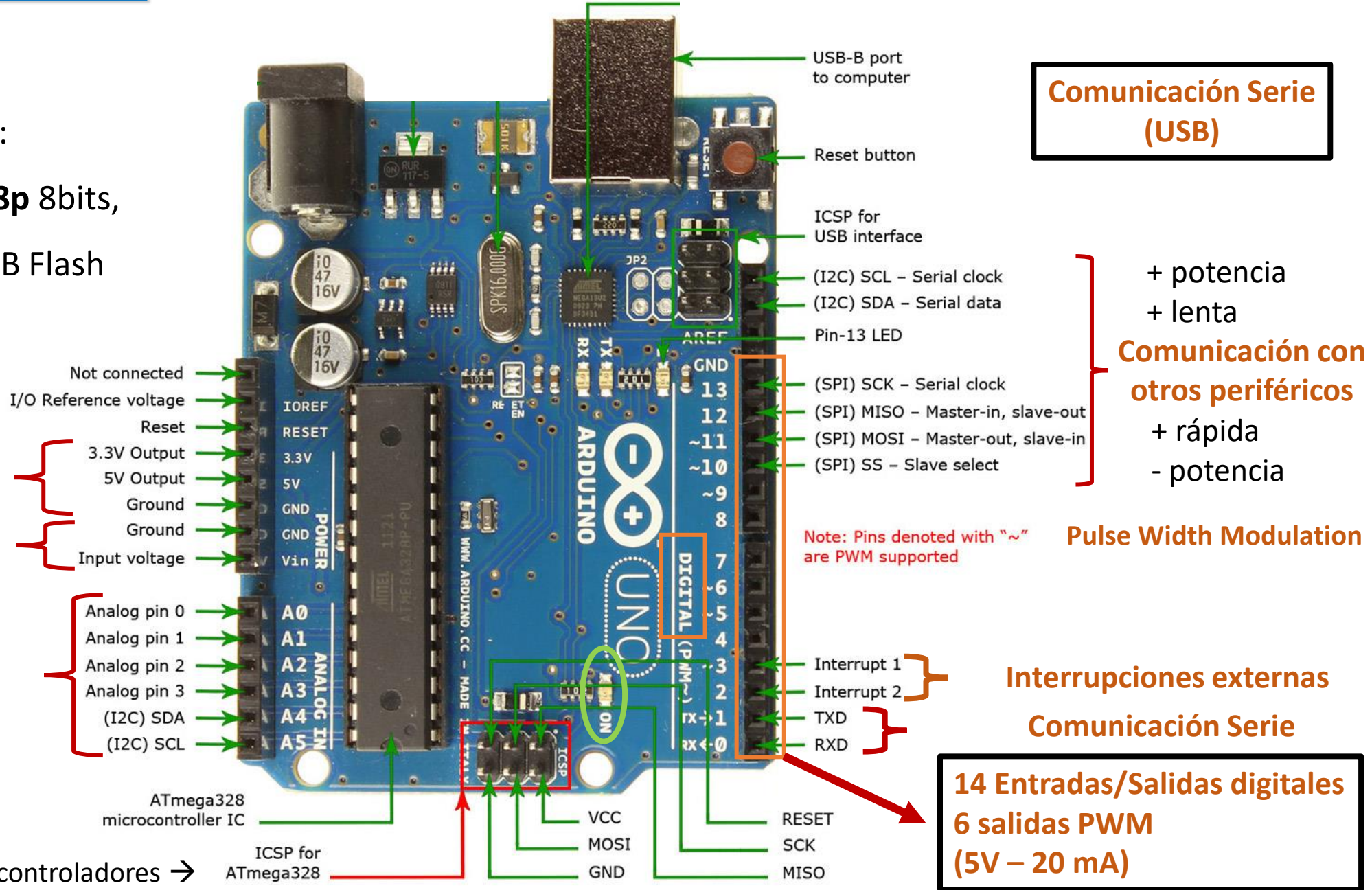
# Arduino UNO

## Pines del microcontrolador

- Microcontrolador:  
**Atmel ATmega328p 8bits,**  
clock: 16MHz 32KB Flash  
( $\tau \sim 60 \text{ ns}$ )

**Alimentación (salida)**

**Entradas analógicas (ADC)**  
 $0 \text{ V} < V_{in} < 5 \text{ V}$

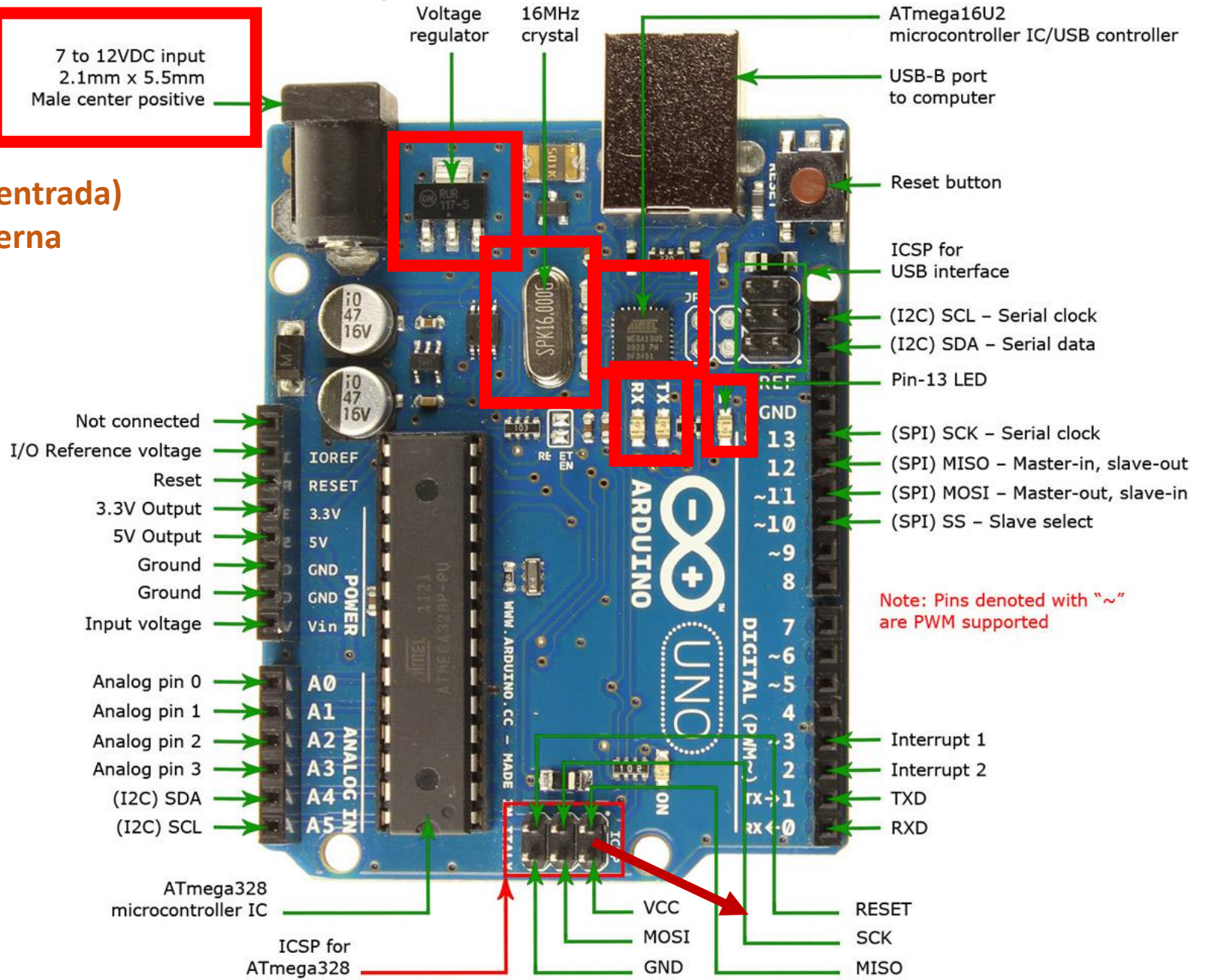


Para programar otros microcontroladores →

# Arduino UNO Elementos sobre la placa

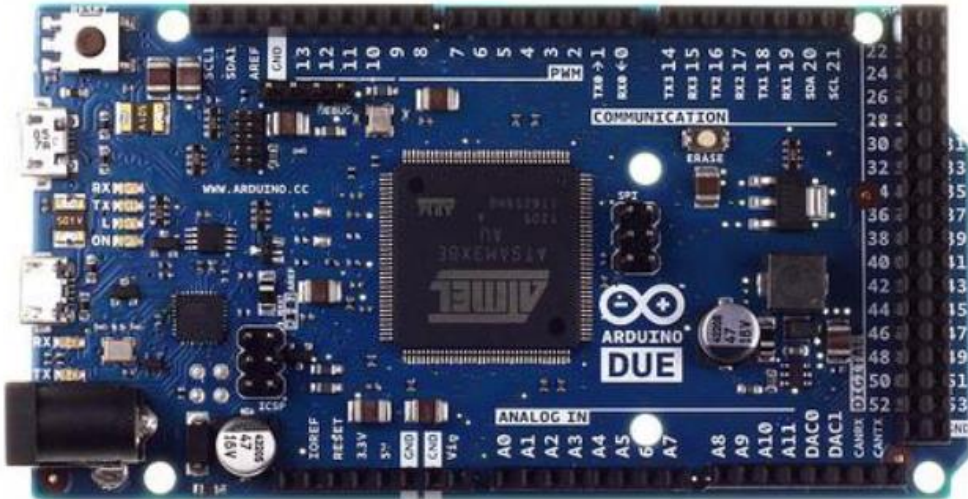
Alimentación (entrada)  
Fuente externa

7 to 12VDC input  
2.1mm x 5.5mm  
Male center positive



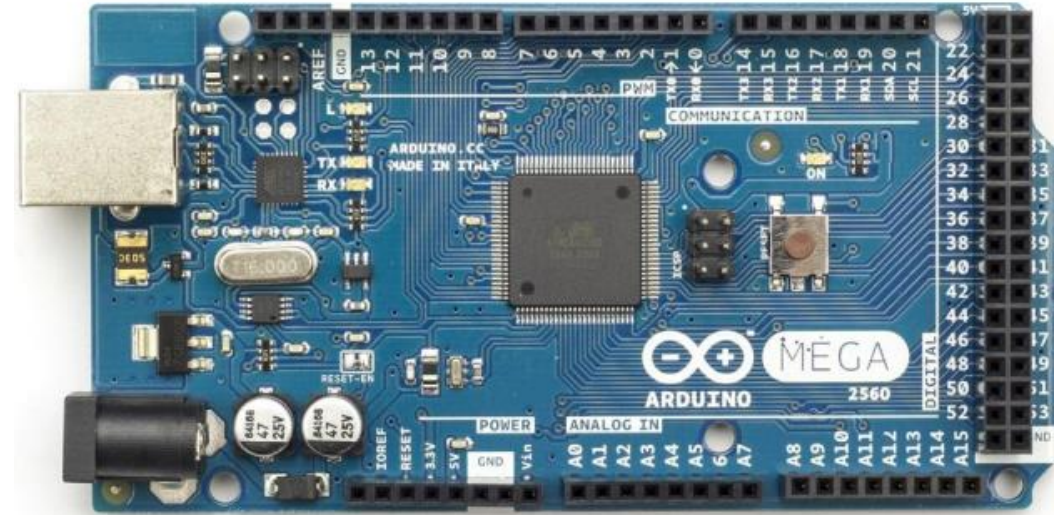
# Otras placas de Arduino

## Due



Microcontrolador de 32 Bits  
Tiene 54 entradas/salidas digitales  
12 entradas analógicas,  
Funcionan todos los módulos basados en 3.3V  
(no soporta 5V)  
2 buses TWI, SPI y 4 UART  
Dos puerto USB para controlar periféricos.

## Mega



Microcontrolador ATmega2560.  
54 entradas/salidas digitales 16 (PWM)  
16 entradas analógicas  
Tiene 6 interrupciones externas.  
4 UART 2 buses modos PWI ,1 SPI

## Nano



- Más barato
- Necesita regulador externo para la tensión de entrada
- Necesita soldar los pines

# Programar Arduino UNO

Programa de placa Arduino



Serial Plot

## Instalación / Programación

**Guía de instalación para Windows y otros**

<https://www.arduino.cc/en/Guide/Windows>

<https://www.arduino.cc/en/Main/Software>

(seguir las instrucciones)

**Guía para la instalación específica de UNO**

<https://www.arduino.cc/en/Guide/ArduinoUno>

**Guía para la programación (en Español)**

<https://www.arduino.cc/reference/es/>

## Adicionales

**Aprendiendo Arduino (Curso)**

<https://aprendiendoarduino.wordpress.com/>

**Algo sobre microcontroladores**

<https://www.newbiehack.com/MicrocontrollersADC10Bits.aspx>

<https://hetpro-store.com/TUTORIALES/adc-del-atmega8/>

# Programar Arduino UNO

1. Conectar Arduino via USB
2. Seleccionar **placa y puerto**

sketch\_sep03a Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda



sketch\_sep03a

```
void setup()
// put your

}

void loop() {
// put your

}
```

- Auto Formato Ctrl+T
- Archivo de programa.
- Reparar codificación & Recargar.
- Administrar Bibliotecas... Ctrl+Mayús+I
- Monitor Serie Ctrl+Mayús+M
- Serial Plotter Ctrl+Mayús+L
- WiFi101 / WiFinINA Firmware Updater
- Placa: "Arduino Uno" >
- Puerto >
- Obtén información de la placa
- Programador: "AVR ISP" >
- Quemar Bootloader

- Gestor de tarjetas...
- Arduino ARM (32-bits) Boards >
- Arduino AVR Boards >

- Arduino Yún
- Arduino Uno
- Arduino Duemilanove or Diecimila
- Arduino Nano
- Arduino Mega or Mega 2560



# Programar Arduino UNO

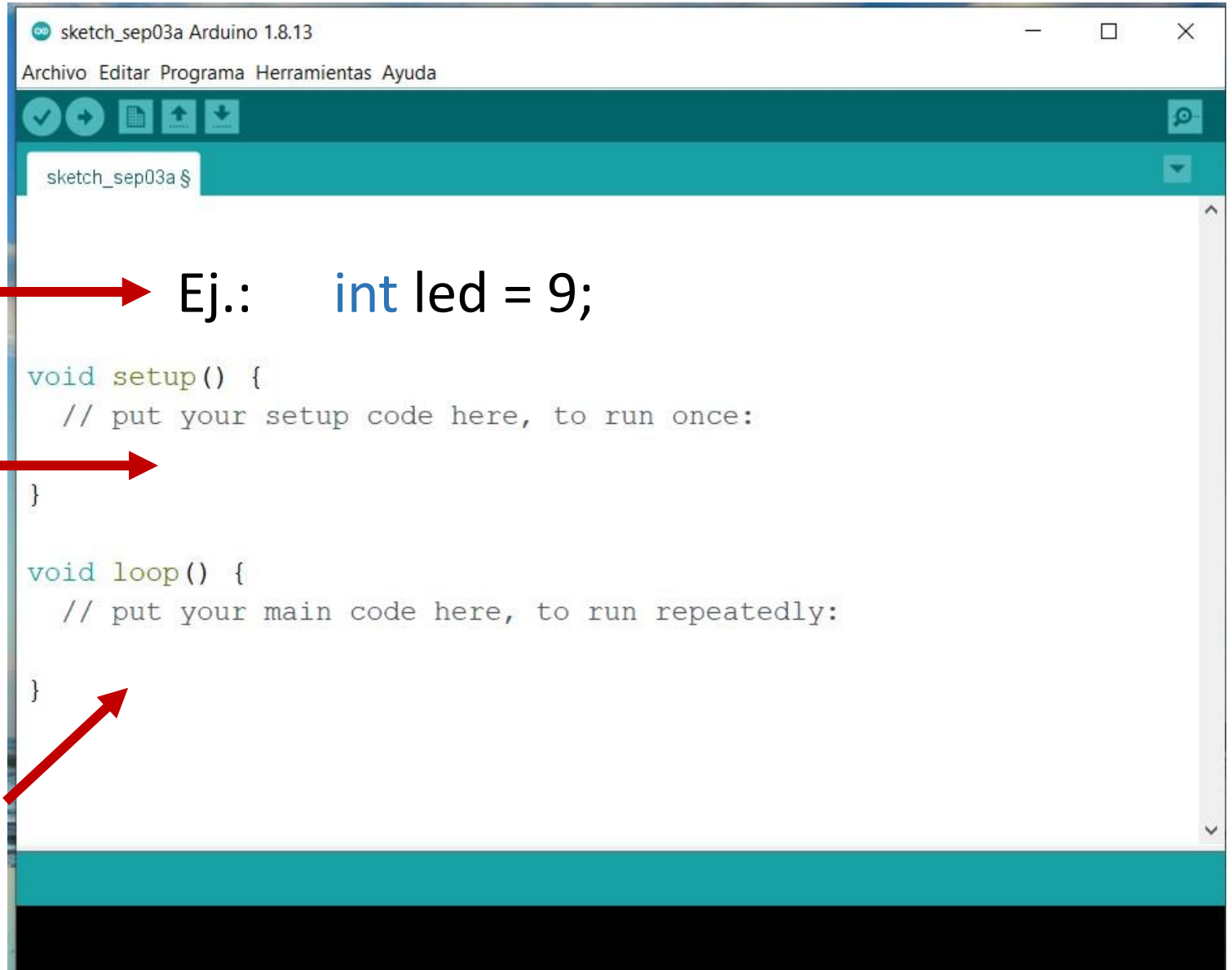
(similar a C++)

¡A diferencia de Python, cuando se crea una variable hay que determinar su tipo!  
Todas las líneas terminan en ; excepto bucles y condicionales

Aquí se declaran las variables globales y su tipo

Aquí se determinan algunos parámetros ligados al microcontrolador, los pines y las comunicaciones

Aquí se especifica la rutina que se ejecutará a modo de Loop



```
sketch_sep03a Arduino 1.8.13
Archivo Editar Programa Herramientas Ayuda
sketch_sep03a $
Ej.: int led = 9;
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

# Tipos de variables

8 bits = 1 byte

`int` → 2 bytes ( $2^{16}$  posibilidades:  $2^{15} \sim 32k$  + signo)

`long` → 4 bytes

`float` → 4 bytes

`double` → en Arduino UNO es igual que float

Las operaciones con enteros (`int`, `long`) son EXACTAS!

`boolean` → 1 byte. `True` o `False`

# Control de tiempos

`delay()` → Tiempo de espera, en milisegundos. Bloquea el código

`millis()` → Milisegundos desde el encendido de la placa

# Comunicación con Arduino UNO: Serial



Void Loop en ejecución

Serial Monitor o Plot

La información entra en una cola (buffer) que se borra cuando se produce la lectura.

Se configura en `void setup() {}`

Configuración:

BAUD (bits por segundo)

data

parity

stop bits

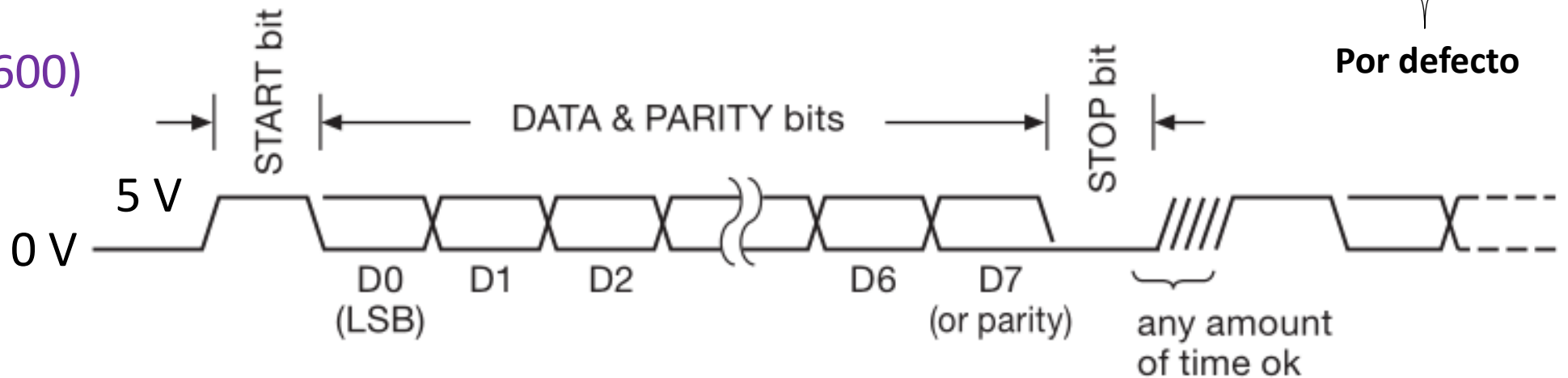
Típ. 9600

8 bits

no

1 stop bit

`Serial.begin(9600)`



# Comunicación con Arduino UNO: Serial

`Serial.print()` → Transforma el contenido en caracteres ASCII (1 byte por caracter)

Ej.: `Serial.print(78)` envía dos bytes con los números 55 y 56

`Serial.println()` → Igual pero agrega un caracter de salto de línea (\n)

`Serial.write()` → Envía un byte con el contenido

`Serial.write(78)` envía un byte con el número 78 (comunicación más rápida)

La pantalla muestra "N" porque Serial Monitor asume ASCII

## ASCII TABLE

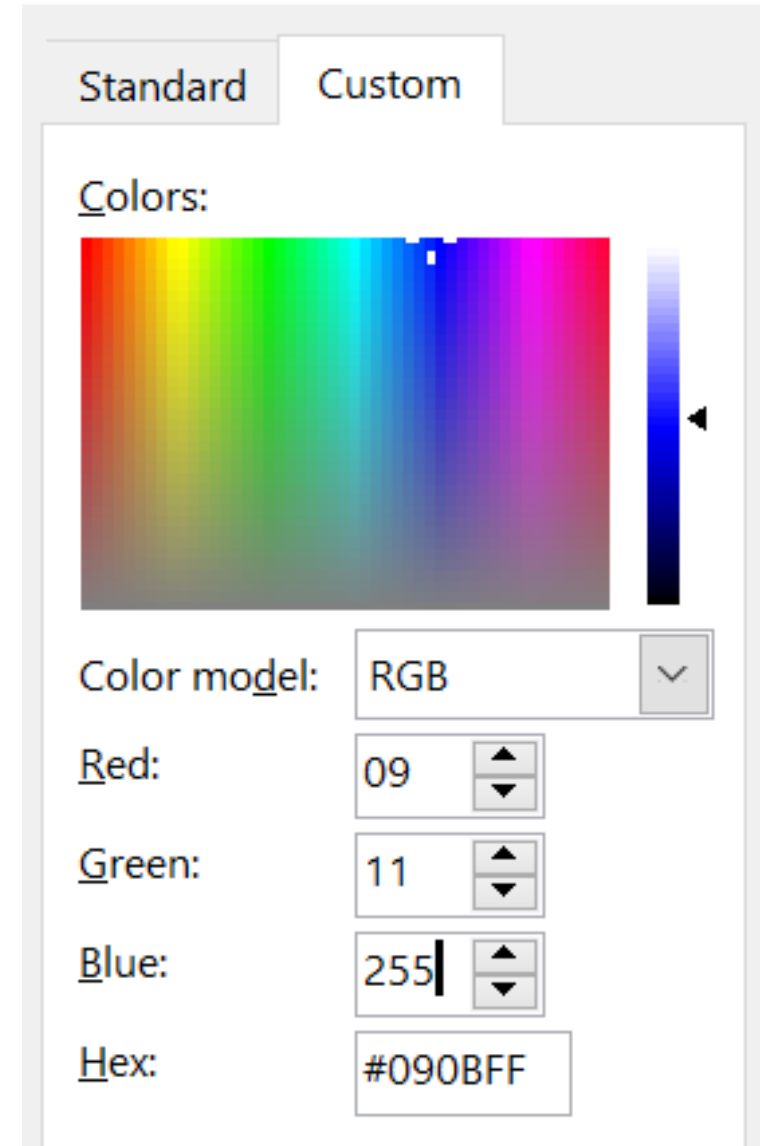
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[DEVICE CONTROL 5]	36	24	\$	68	44	D	100	64	d
5	5	[DEVICE CONTROL 6]	37	25	%	69	45	E	101	65	e
6	6	[DEVICE CONTROL 7]	38	26	&	70	46	F	102	66	f
7	7	[DEVICE CONTROL 8]	39	27	'	71	47	G	103	67	g
8	8	[DEVICE CONTROL 9]	40	28	(	72	48	H	104	68	h
9	9	[DEVICE CONTROL 10]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[DEVICE CONTROL 11]	43	2B	+	75	4B	K	107	6B	k
12	C	[DEVICE CONTROL 12]	44	2C	,	76	4C	L	108	6C	l
13	D	[DEVICE CONTROL 13]	45	2D	-	77	4D	M	109	6D	m
14	E	[DEVICE CONTROL 4]	46	2E	.	78	4E	N	110	6E	n
15	F	[NEGATIVE ACKNOWLEDGE]	47	2F	:	79	4F	O	111	6F	o
16	10	[SYNCHRONOUS IDLE]	48	30	;	80	50	P	112	70	p
17	11	[ENG OF TRANS. BLOCK]	49	31	<	81	51	Q	113	71	q
18	12	[CANCEL]	50	32	=	82	52	R	114	72	r
19	13	[DEVICE CONTROL 17]	51	33	>	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x

# Representación hexadecimal

- Es útil para representar un byte.
- 1 byte = 8 bit  $\rightarrow 2^8 = 256$  posibilidades:
  - 4 bit  $\rightarrow 2^4 = 16$  posibilidades:
    - 0, 1, 2, ..., 9, A, B, C, D, E, F
  - 2 hex =  $16^2 = 256$  posibilidades
- Ejemplo: Uso en representación de colores (256 colores por canal RGB)

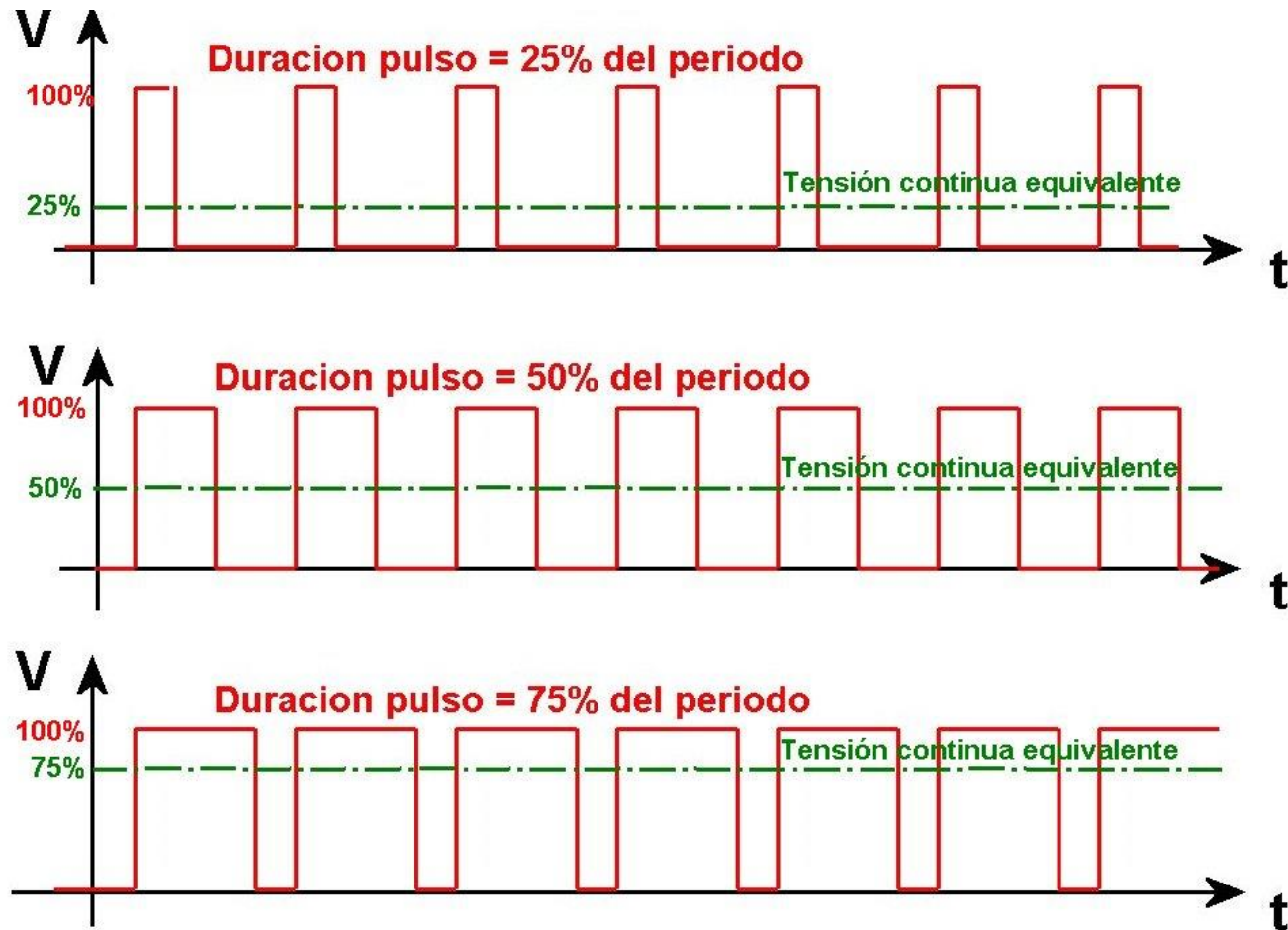
DECIMAL	9	11	255
HEX	09	0B	FF

## Colors



# Control por Ancho de Pulso PWM (pulse width modulation)

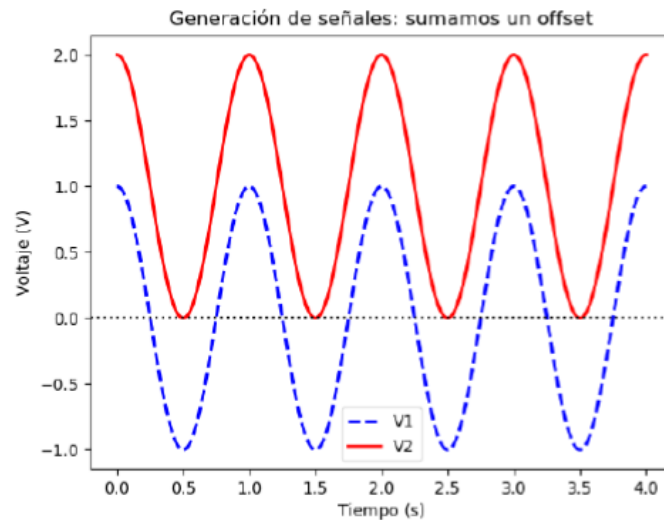
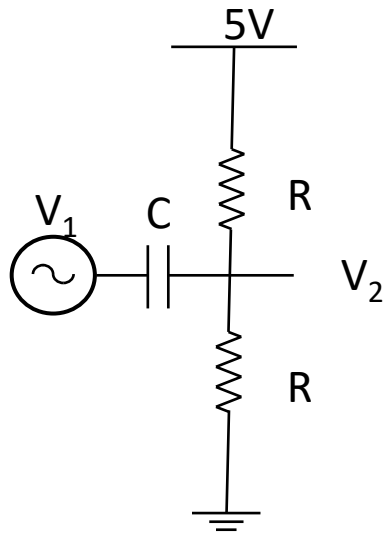
Se varia el ciclo de trabajo (Duty Cycle). Hay dos periodos fijos para elegir.



En dispositivos con respuesta temporal lenta, permite regular intensidades de luz, velocidades de giro, mediante una señal digital.

# Acondicionamiento de señal utilizando un divisor resistivo

**Arduino** mide solo señales positivas. Entonces hay que sumar un offset a señales  $< 0$  V. ¿Cómo lo hacemos?



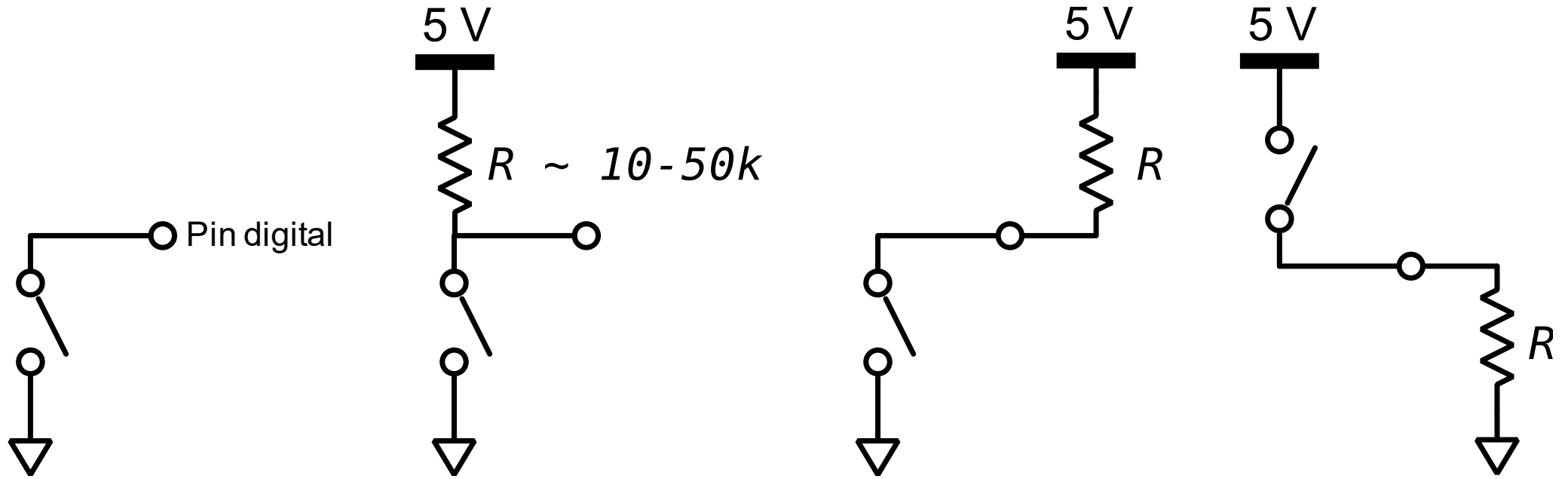
Una señal de 2 Vpp va entre 1 y -1 V

Acondicionada se mueve entre 0 V y 2 V

Ahora es posible medirla con Arduino

# Resistencias pull-up o pull-down

El estado lógico (tensión) de una entrada desconectada no está definido.



INDEFINIDO

PULL-UP EXTERNO

PULL-UP INTERNO

PULL-DOWN INTERNO



# Control de Arduino con Python



Open port at “9600,8,N,1”, no timeout:

<https://pythonhosted.org/pyserial/index.html>

```
>>> import serial
>>> ser = serial.Serial('/dev/ttyUSB0') # open serial port
>>> print(ser.name)                    # check which port was really used
>>> ser.write(b'hello')                # write a string
>>> ser.close()                        # close port
```

Open named port at “19200,8,N,1”, 1s timeout:

```
>>> with serial.Serial('/dev/ttyS1', 19200, timeout=1) as ser:
...     x = ser.read()                 # read one byte
...     s = ser.read(10)               # read up to ten bytes (timeout)
...     line = ser.readline()         # read a '\n' terminated line
```

# Algunos sensores

**Sensor ultrasonido**



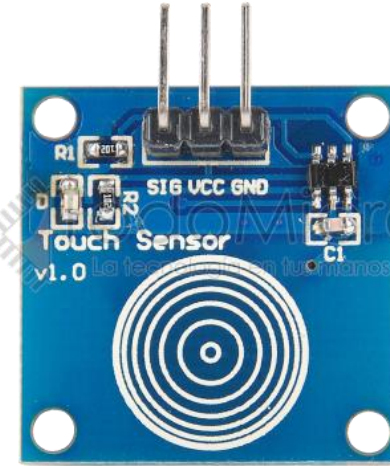
**Temperatura y Humedad**



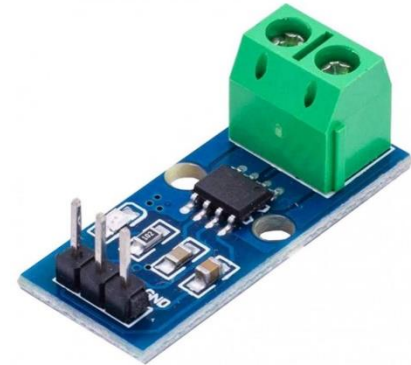
**Sensor de movimiento**



**Sensor Touch capacitivo**



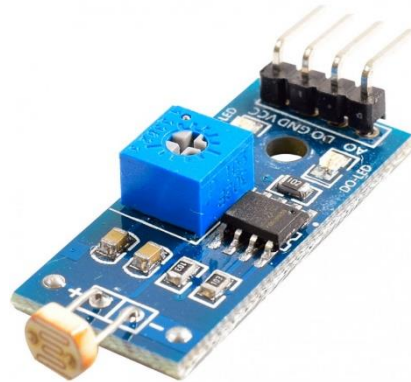
**Sensor corriente (efecto hall)**



**Sensor monóxido de carbono**



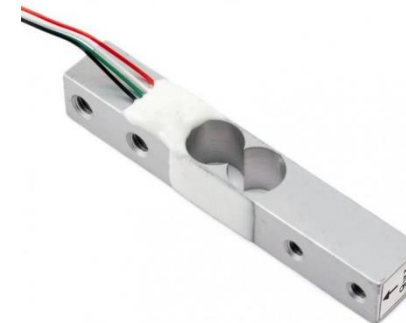
**Sensor Luz Fotoresistor**



**Sensor Humedad Suelo**



**Celda de carga**



**Sensor intensidad de luz**

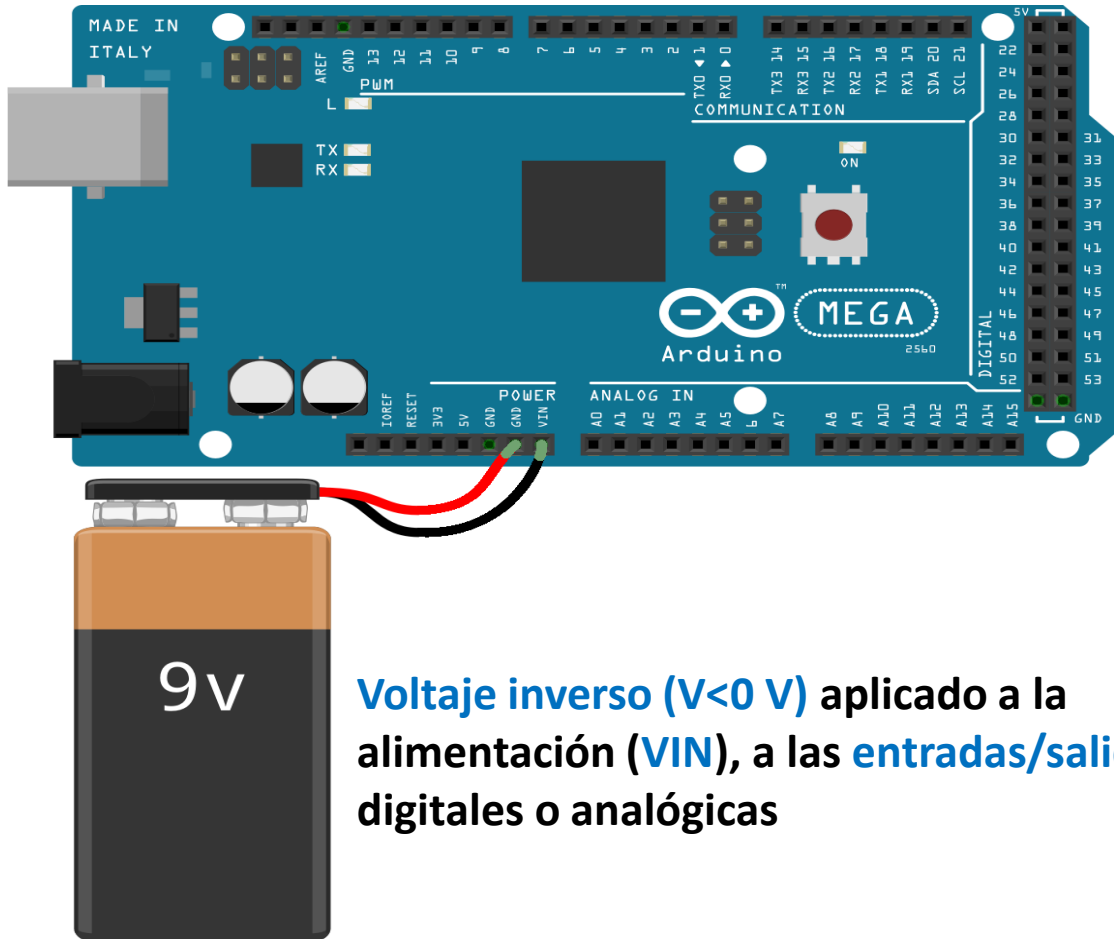


# Actividades

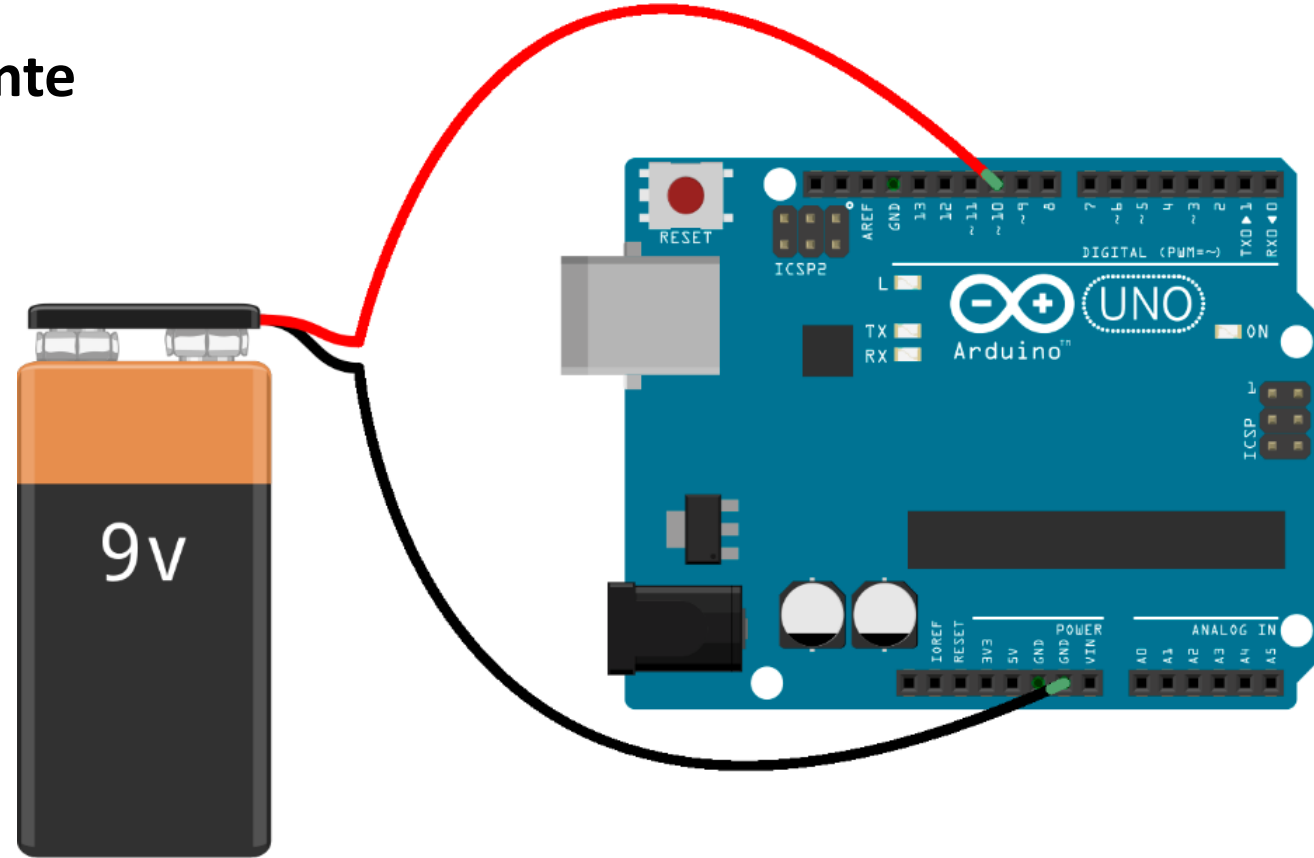
1. Encender un LED externo cada cierto tiempo. Limitar la corriente  $< 20$  mA!
  1. *Blink.ino*
  2. *Blink without delay.ino*
2. Leer una señal analógica con un potenciómetro
  1. *analogInput.ino*
3. Controlar intensidad de LED usando una señal PWM + potenciómetro
  1. Usando `analogWrite()`
4. Medir con el osciloscopio el tren de bits en la comunicación Serie e interpretar.
5. Acondicionar una señal del generador de funciones para poder medirla con Arduino.

# Cómo destruir un Arduino...o todo lo que hay que evitar!!!

- Voltaje inverso
- Exceso de corriente
- Sobrevoltaje
- Otras...

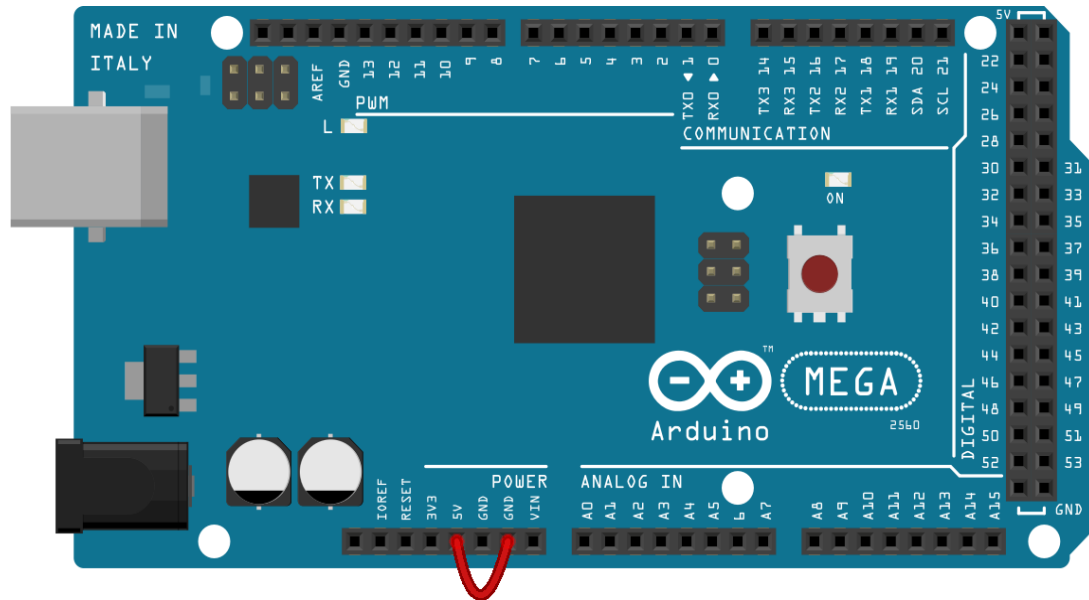


**Voltaje inverso ( $V < 0\text{ V}$ )** aplicado a la alimentación (VIN), a las **entradas/salidas digitales** o analógicas

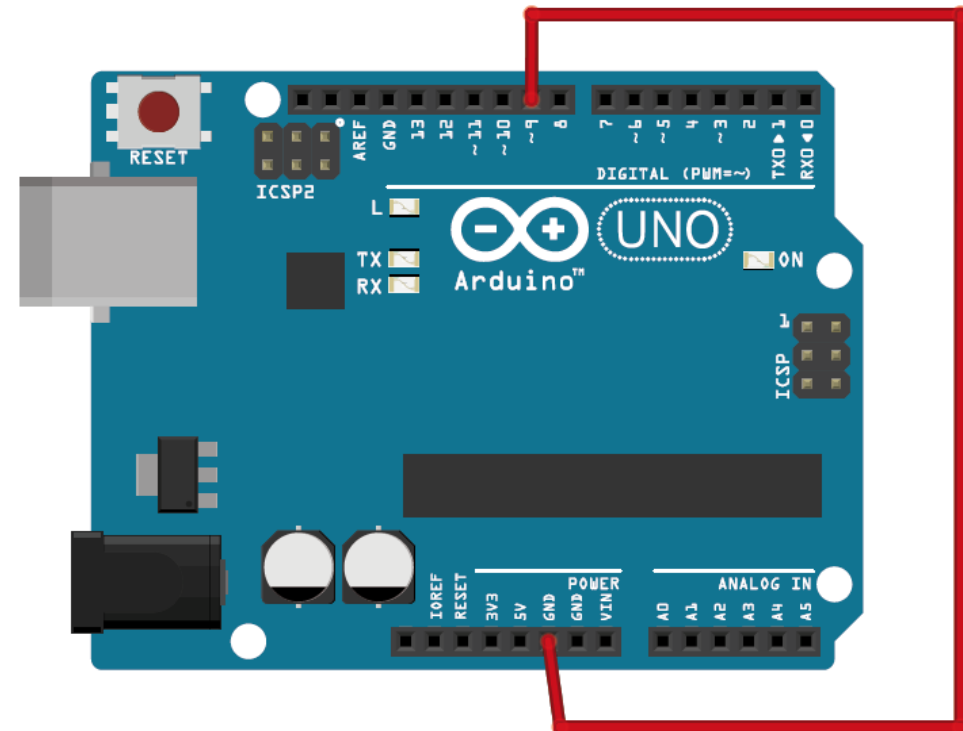


**Sobrevoltaje ( $V > 5\text{ V}$ )** aplicado a un pin **analógico/digital** o a un pin de **alimentación (VIN)**

# Cómo destruir un Arduino...o todo lo que hay que evitar!!!



**Cortocircuito en placa!**



**Sobrecorriente en pin digital (> 20 mA)**  
Para evitarla, usar siempre una  $R \geq 220$  ohm para este tipo de conexión

# Algunos ejemplos Arduino UNO

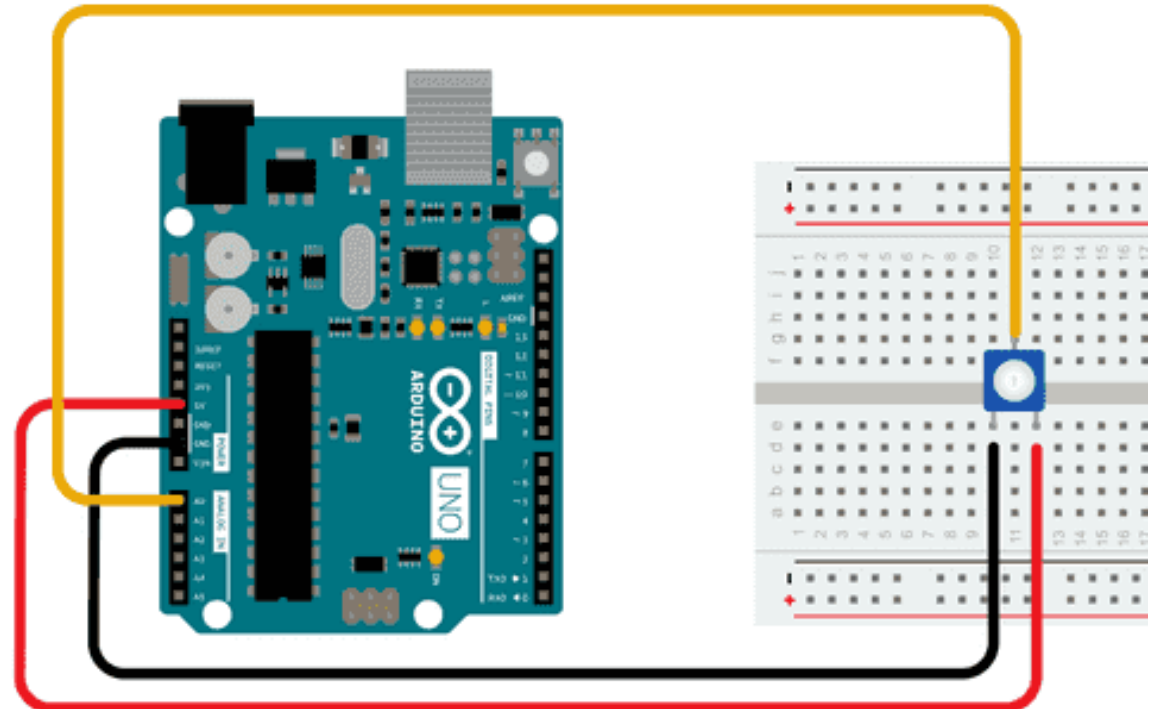
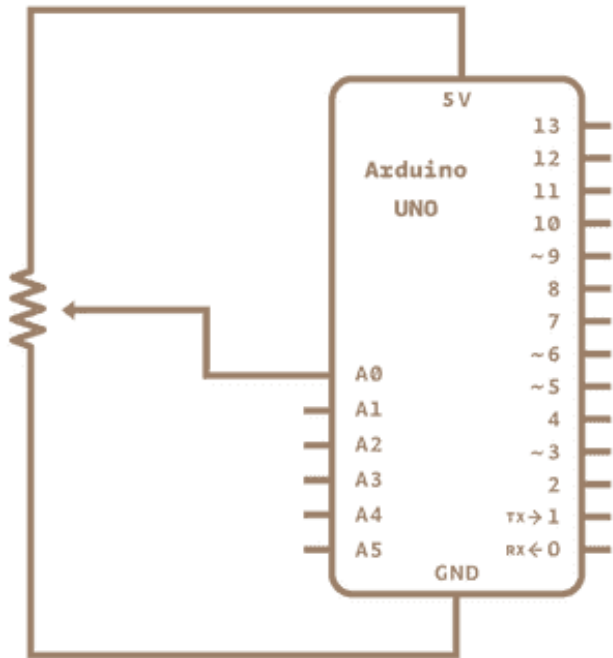
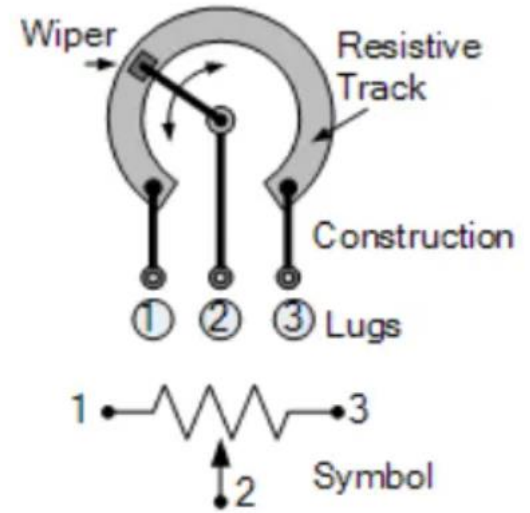
<https://docs.arduino.cc/built-in-examples/>

- **Leer una señal analógica**

Conectar la salida de un potenciómetro a una entrada analógica

Permitira poner la entrada A<sub>0</sub> a una tensión intermedia entre 0 y 5 V.

Utilizar un potenciómetro de 10 kΩ



## Leer una señal Analógica

```
AnalogReadSerial

Reads an analog input on pin 0, prints the result to the Serial Monitor.
Graphical representation is available using Serial Plotter (Tools > Serial Pl
Attach the center pin of a potentiometer to pin A0, and the outside pins to +

This example code is in the public domain.

https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogReadSerial
*/

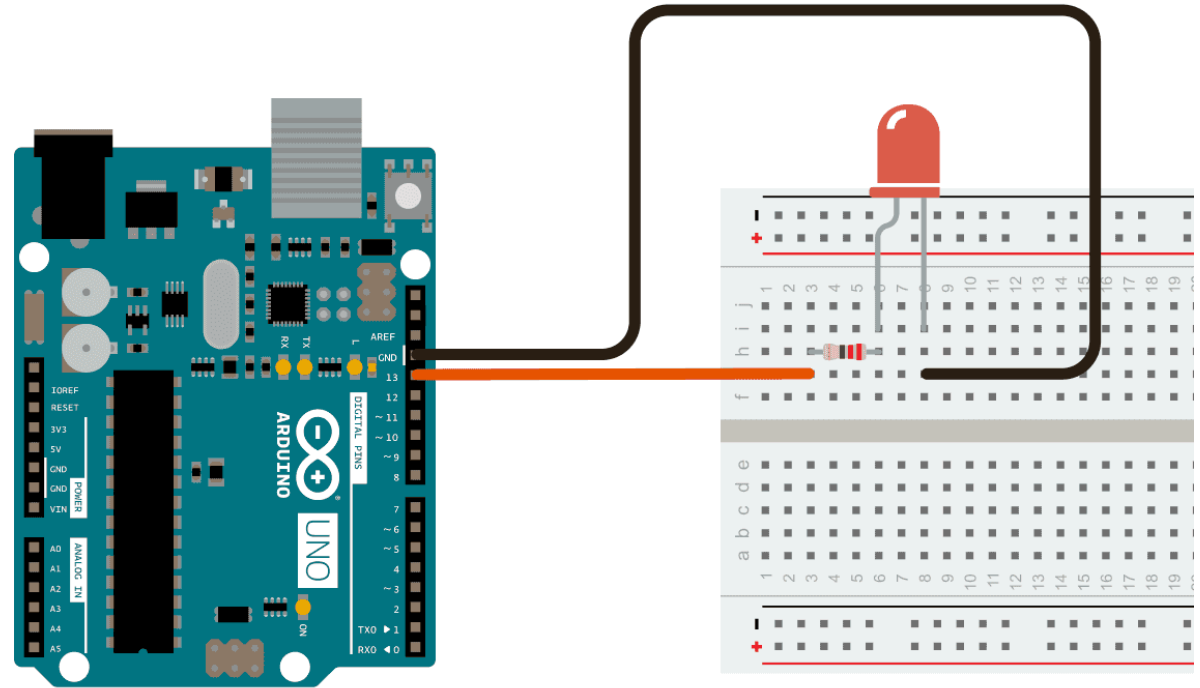
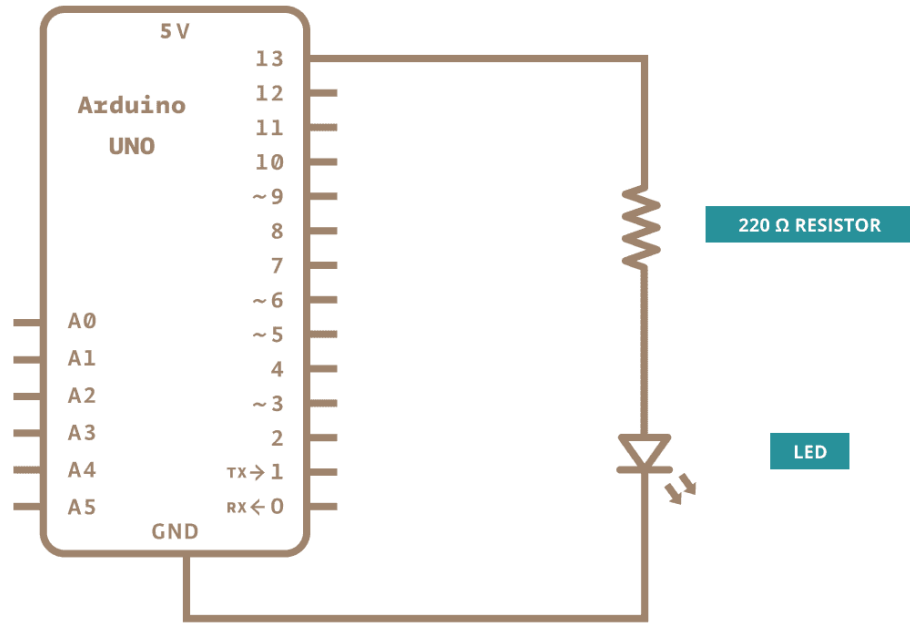
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1); // delay in between reads for stability
}
```

## Leer una tensión

```
// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V)
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

# Destello de un LED Blink

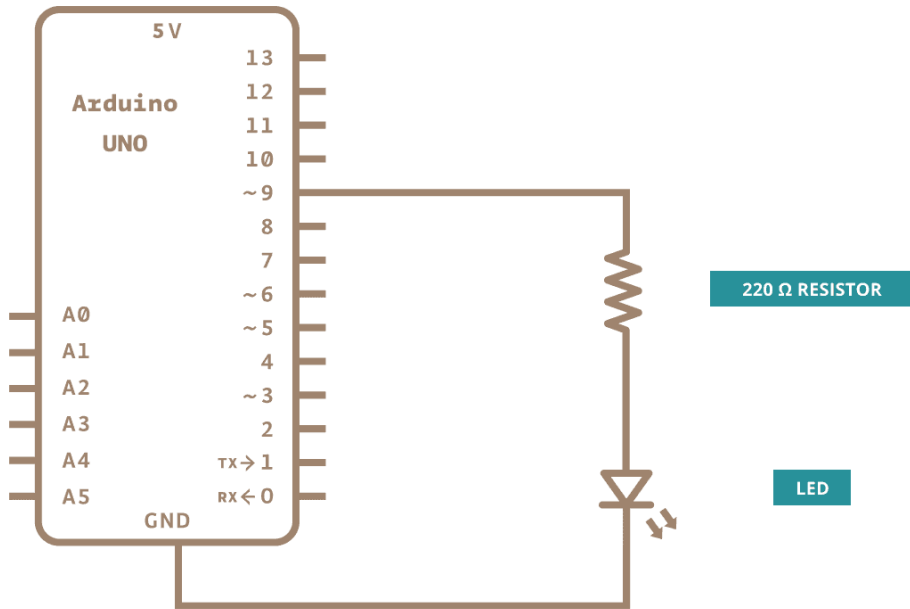


```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```



# Desvanecimiento de un LED Fading a LED Mediante PWM Salidas digitales con ~



This example code is in the public domain.

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/Fade>  
\*/

```
int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```