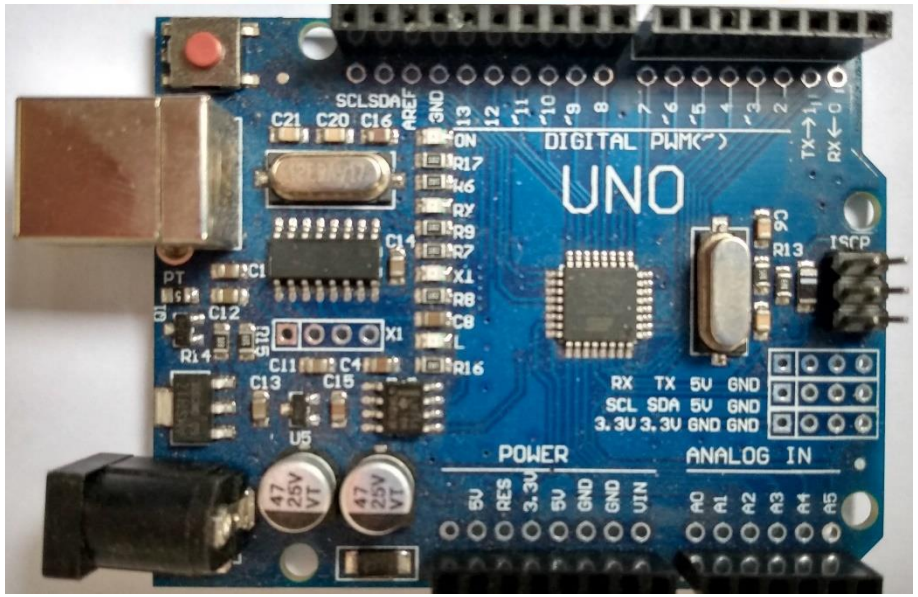
ARDUINO

LABORATORIO 3 - DF - FCEyN - UBA

2do cuatrimestre de 2022

**Arduino es una plataforma de creación de electrónica de software y código abierto**, la cual está basada en una placa con todos los elementos necesarios para **conectar periféricos a las entradas y salidas de un microcontrolador**, y que puede ser programada en software libre, flexible y fácil de utilizar: Arduino IDE (Entorno de Desarrollo Integrado)

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.



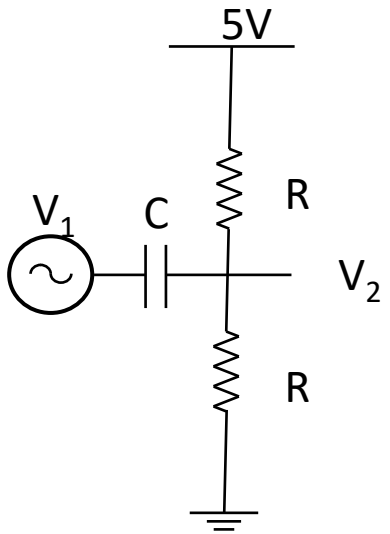

# Arduino Uno

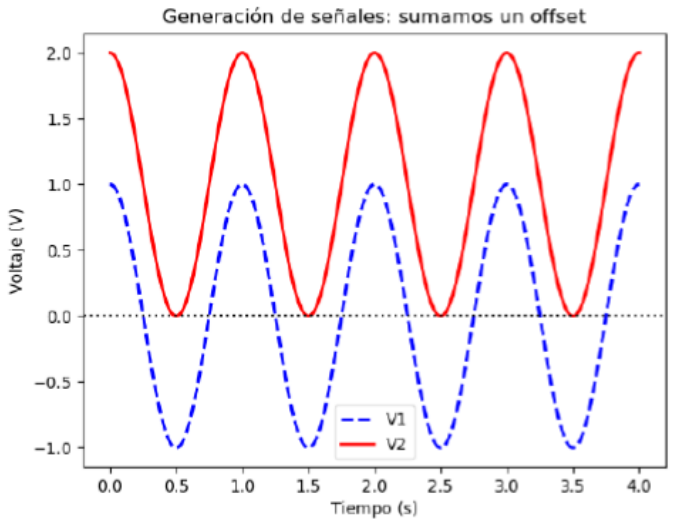
Precio Aproximado 4000$


- 6 Entradas analógicas Rango [0,5V]
- 16 canales digitales (Entrada/Salida) ~ 6 para PWM
- Conversor Analógico digital 10 Bits
- 2 entradas para interrupciones
- Fuentes 3.3V y 5V
- Microcontrolador Atmel ATmega328p 8bits, clock: 16MHz 32KB Flash

**Obs:** Mide solo señales positivas, será necesario sumar un offset a las generadas con la placa de sonido. Para poder medirlas enteras

<u>Acondicionamiento de señal utilizando un divisor resistivo</u>



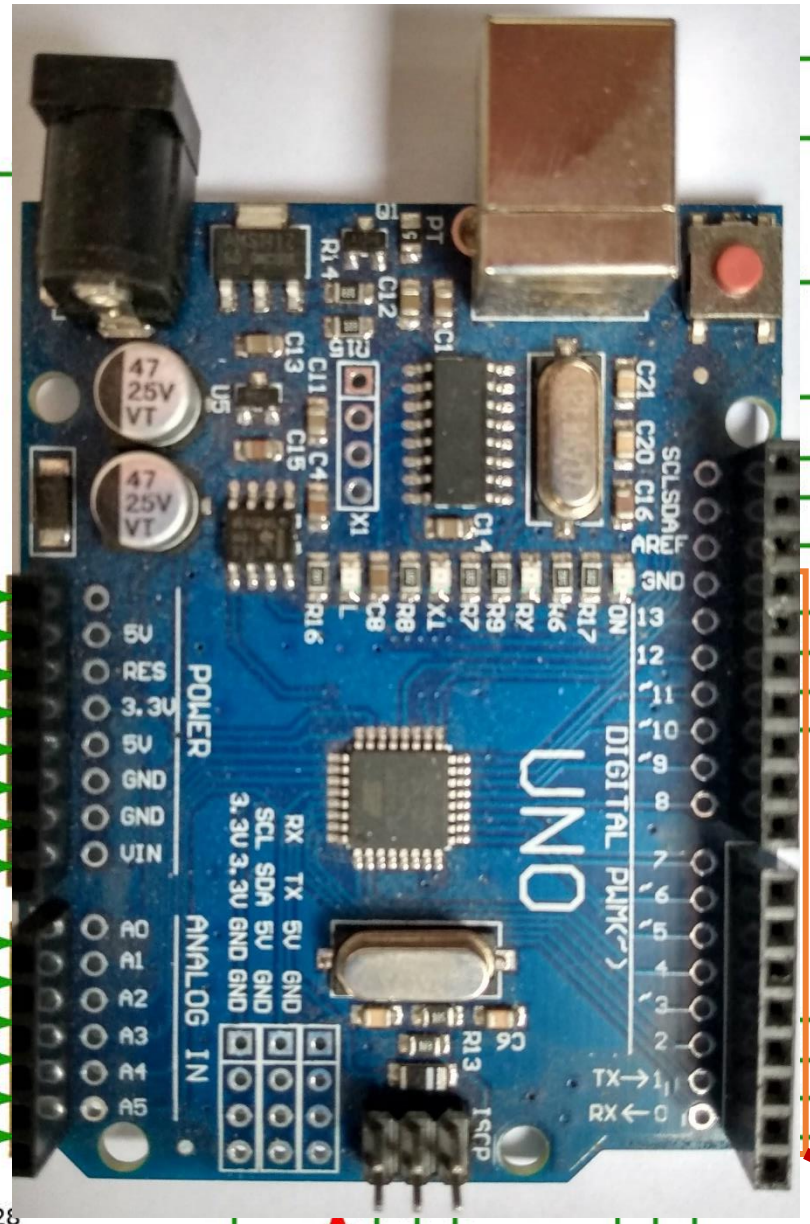




Una señal de 2 Vpp va entre 1 y -1v

Acondicionada se mueve entre 0v y 2v

Ahora es posible medirla con Arduino

# Arduino UNO




**Alimentación (entrada)**
**Fuente externa**

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

ATmega16U2
microcontroller IC/USB controller

USB-B port
to computer

**Comunicación Serie (USB)**

Reset button

ICSP for
USB interface

(I2C) SCL – Serial clock
(I2C) SDA – Serial data
Pin-13 LED

(SPI) SCK – Serial clock
(SPI) MISO – Master-in, slave-out
(SPI) MOSI – Master-out, slave-in
(SPI) SS – Slave select

+ potencia
+ lenta
**Comunicación con otros periféricos**
+ rápida
- potencia

Not connected
I/O Reference voltage
Reset
3.3V Output
5V Output
Ground
Ground
Input voltage

**Alimentación (salida)**

**Alimentación (entrada) x ej. pila**

Analog pin 0
Analog pin 1
Analog pin 2
Analog pin 3
(I2C) SDA
(I2C) SCL

**Entradas analógicas (ADC)**
**0 V<Vin<5 V**

Note: Pins denoted with "~" are PWM supported

**Pulse Width Modulation**

Interrupt 1
Interrupt 2

**Interrupciones externas**

TXD
RXD

**Comunicación Serie**

**14 Entradas/Salidas digitales**
**6 salidas PWM**
**(5V – 20 mA)**

ATmega328
microcontroller IC

ICSP for
ATmega328

VCC
MOSI
GND

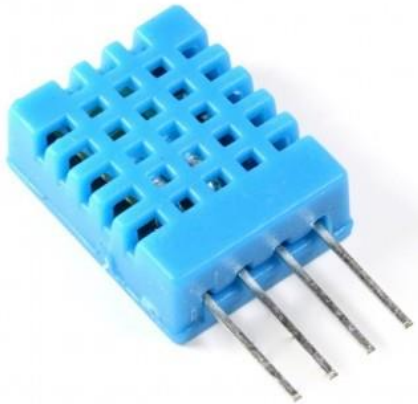
RESET
SCK
MISO

Para programar otros microcontroladores →

# Driver y motores

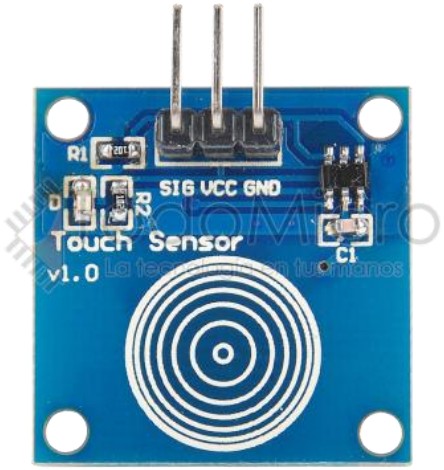
Sensor ultrasonido


Temperatura y Humedad



Sensor de movimiento

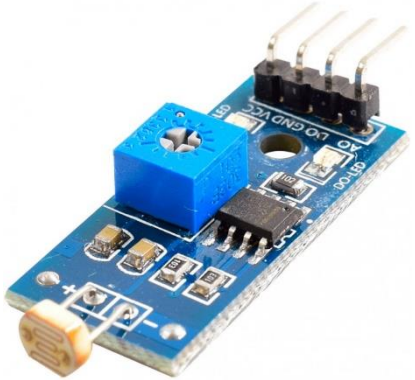
Sensor Touch capacitivo




Sensor corriente (efecto hall)

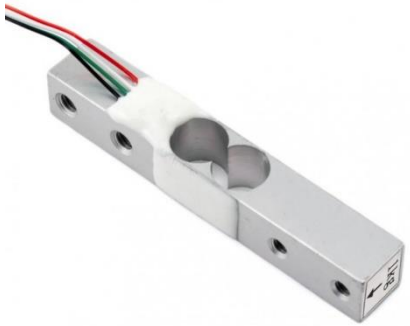
Sensor monóxido de carbono

Sensor Luz Fotoresistor

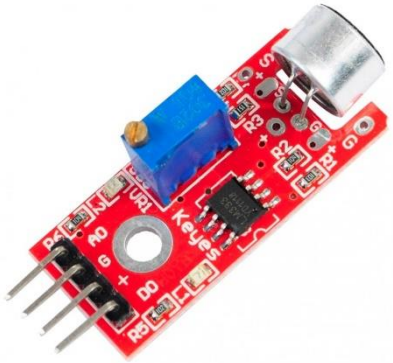


Sensor Humedad Suelo

Celda de carga

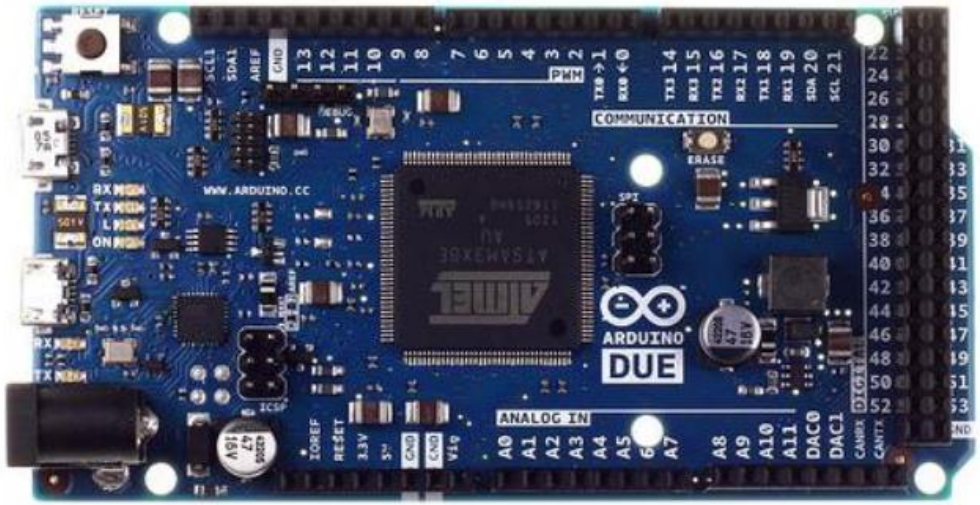


Sensor intensidad de luz



# Tipos de arduino

**Due**

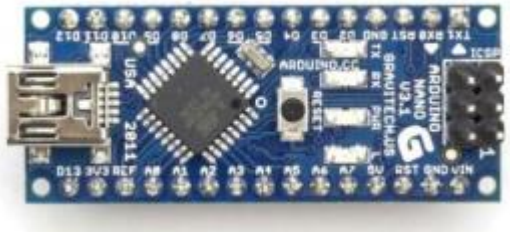



Microcontrolador de 32 Bits
Tiene 54 entradas/salidas digitales
12 entradas analógicas,
Funcionan todos los módulos basados en 3.3V (no soporta 5V)
2 buses TWI, SPI y 4 UART
Dos puerto USB para controlar periféricos.

**Mega**

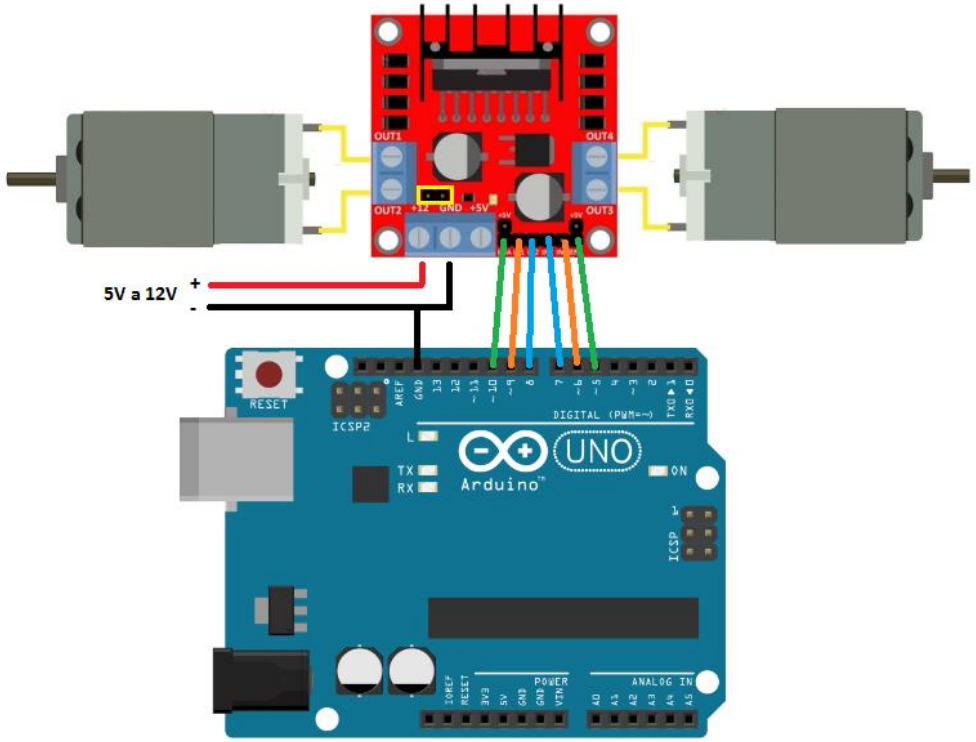

Microcontrolador ATmega2560.
54 entradas/salidas digitales 16 (PWM)
16 entradas analógicas
Tiene 6 interrupciones externas.
4 UART 2 buses modos PWI ,1 SPI

**Nano**



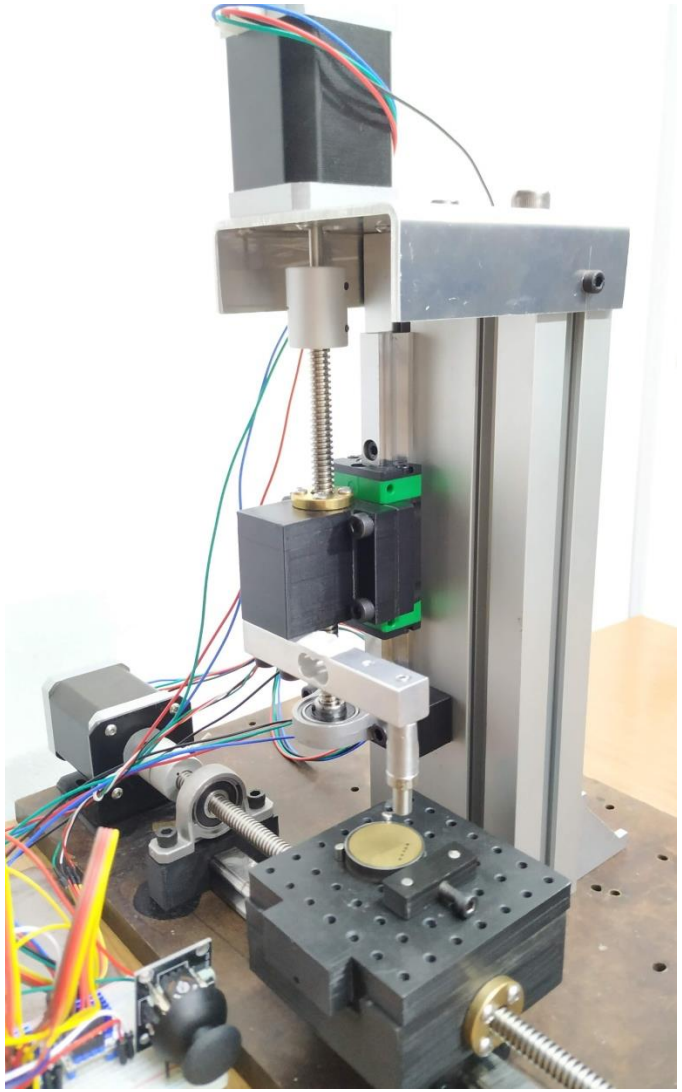
Parecido al arduino uno, mas pequeño es necesario soldarle los terminales

# Drivers y motores




ESP 23

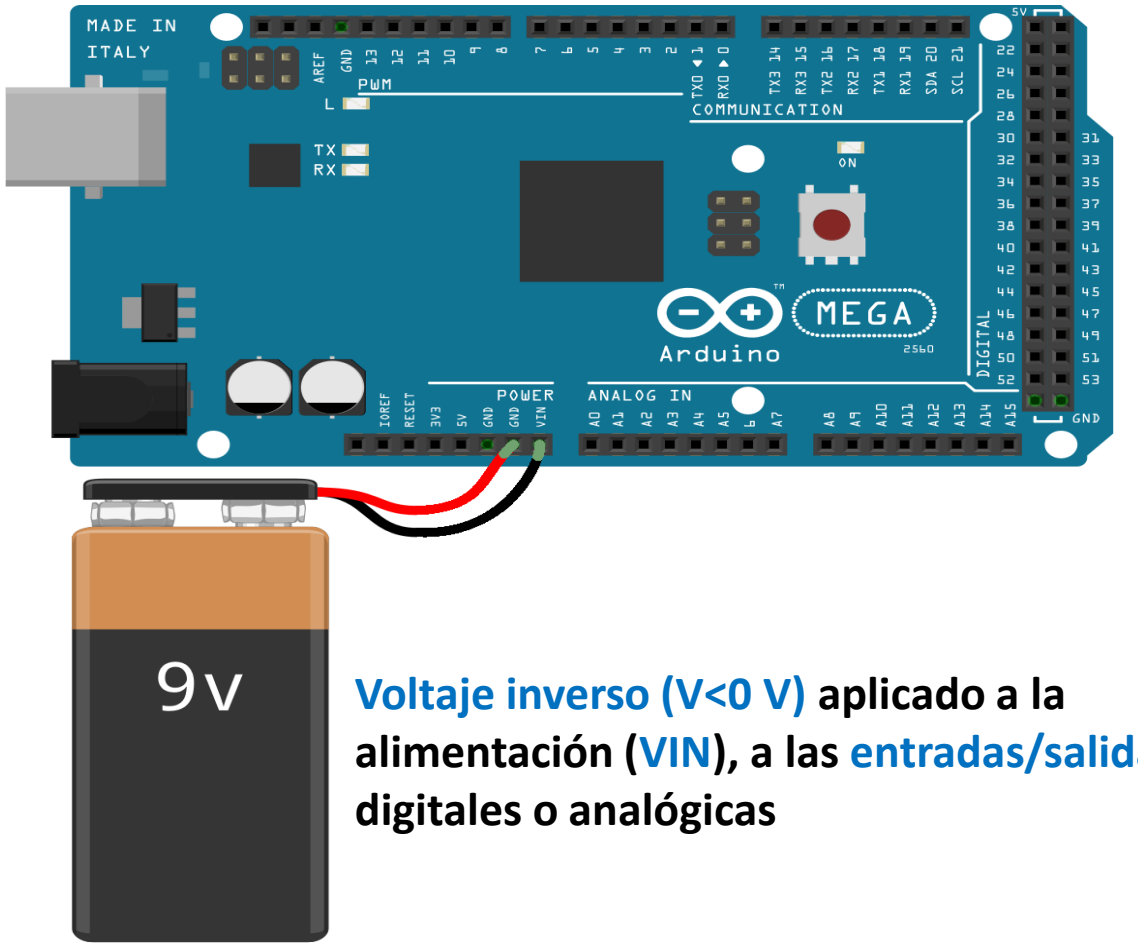
Equipo de rayado



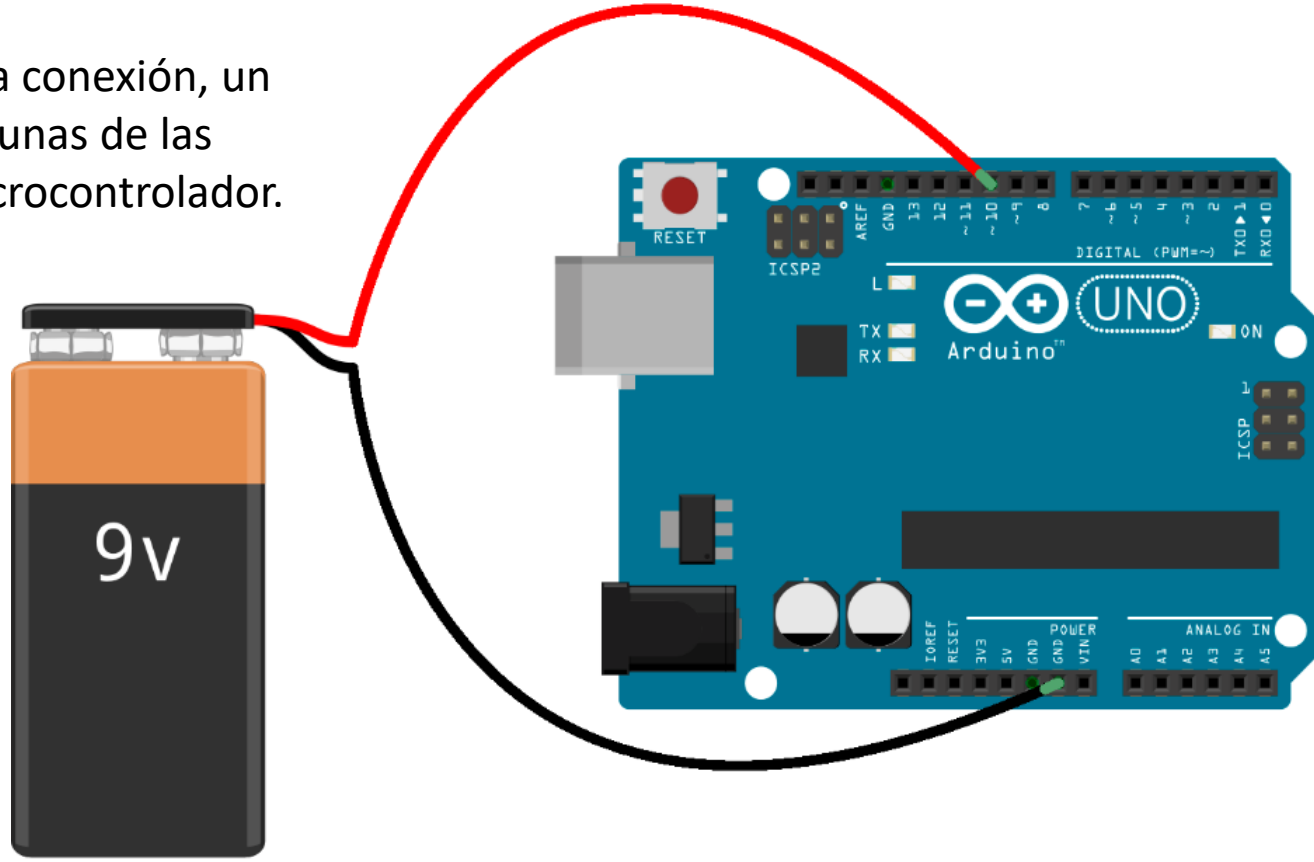

2 microprocesadores
Clock 240MHz
Wifi

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

Existen muchas formas de destruir un Arduino. Una mala conexión, un sobrevoltaje o un exceso de corriente son solamente algunas de las principales razones que llevan a la destrucción de un microcontrolador.



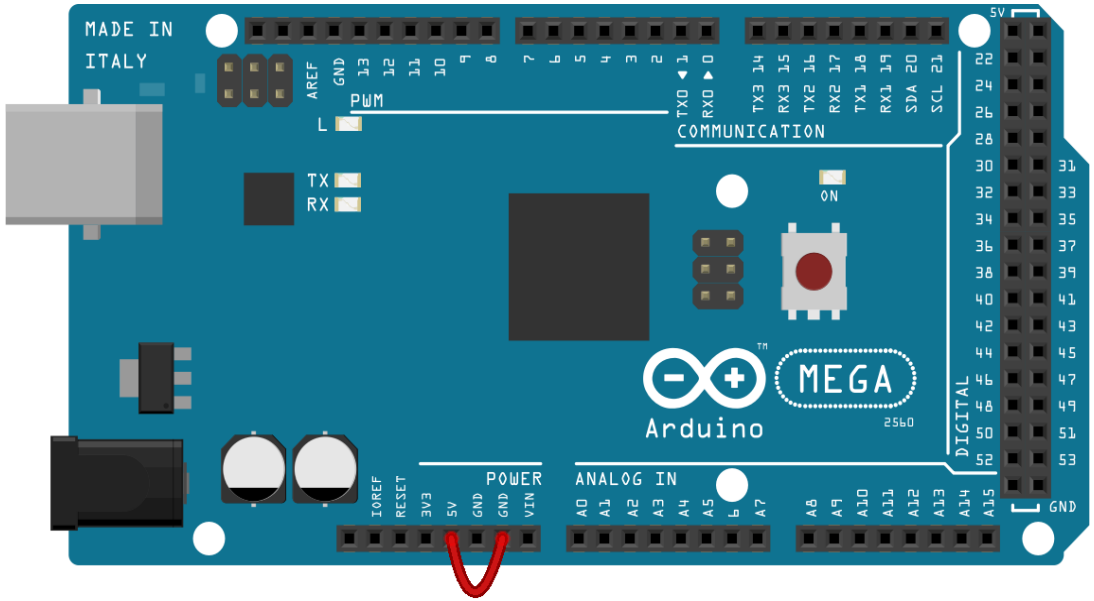

**Voltaje inverso (V<0 V) aplicado a la alimentación (VIN), a las entradas/salidas digitales o analógicas**



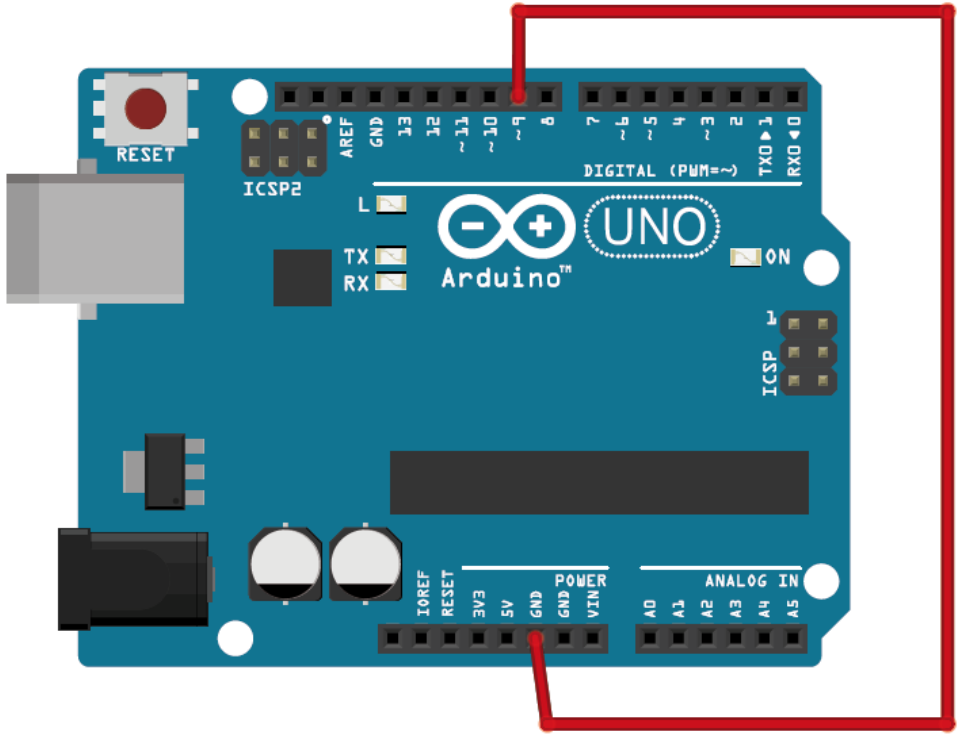

**Sobrevoltaje (V>5 V) aplicado a un pin analógico/digital o a un pin de alimentación (VIN)**

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

Existen muchas formas de destruir un Arduino. Una mala conexión, un sobrevoltaje o un exceso de corriente son solamente algunas de las principales razones que llevan a la destrucción de un microcontrolador.




**Cortocircuito en placa!**




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

# 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/

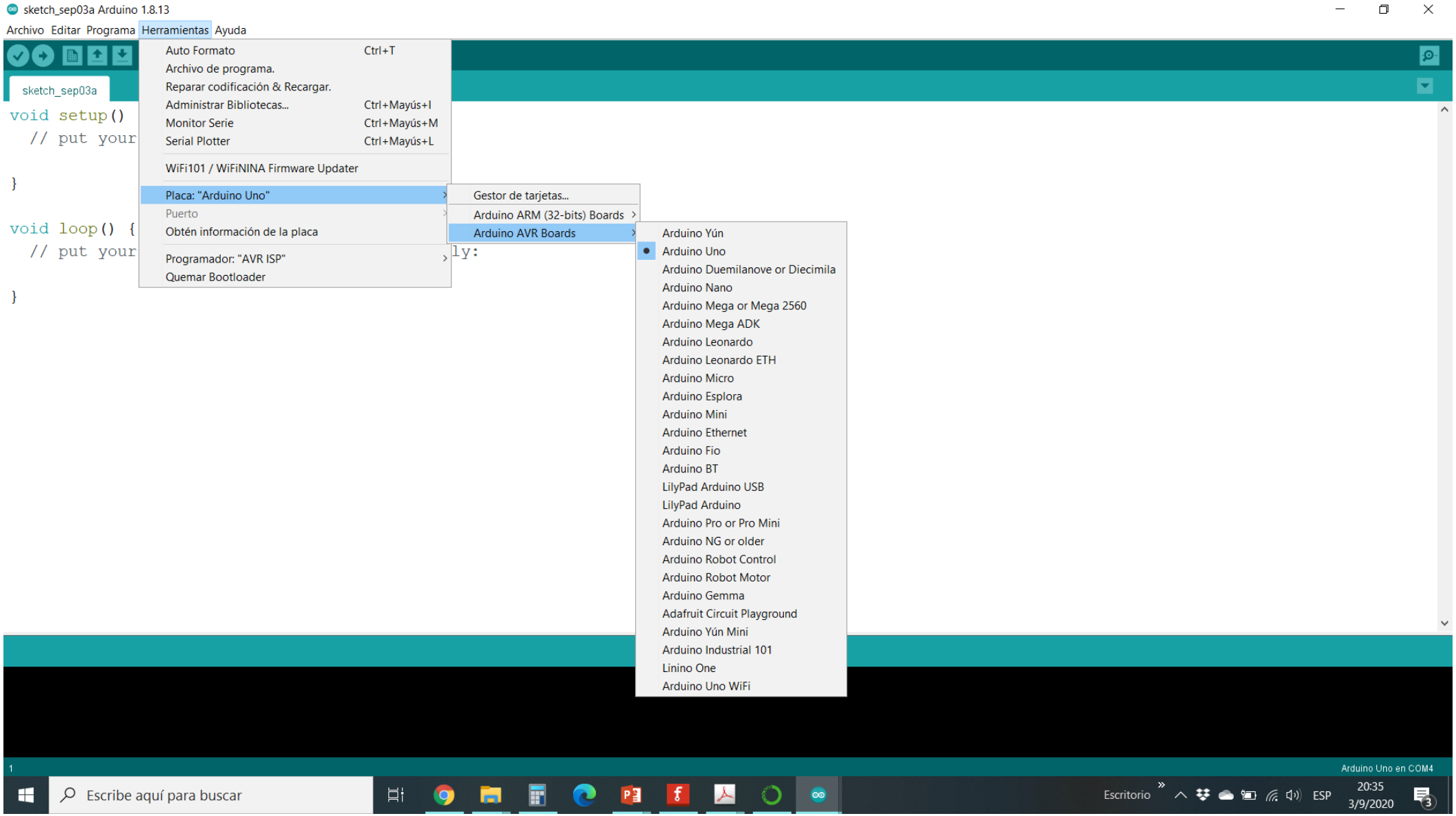
## Adicionales

**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



# Programar Arduino UNO

Programación en C:

Aquí se declaran las variables (globales) y su tipo

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

sketch_sep03a Arduino 1.8.13

Archivo Editar Programa Herramientas Ayuda

sketch_sep03a §

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

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

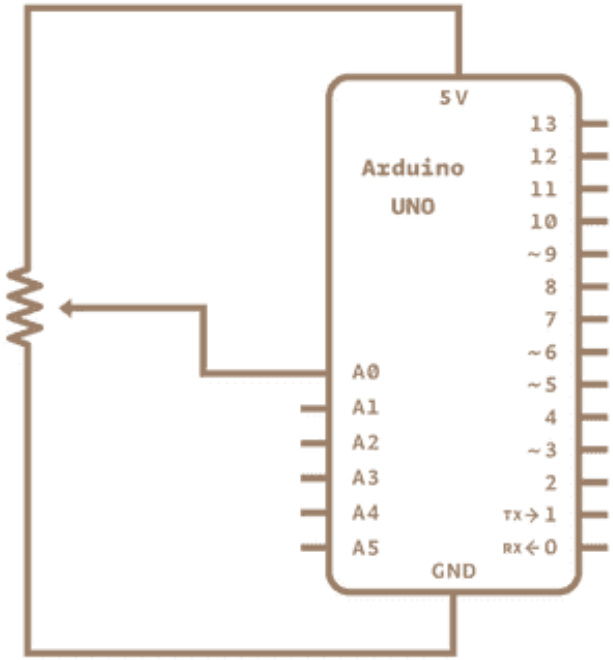
# Algunos ejemplos Arduino UNO

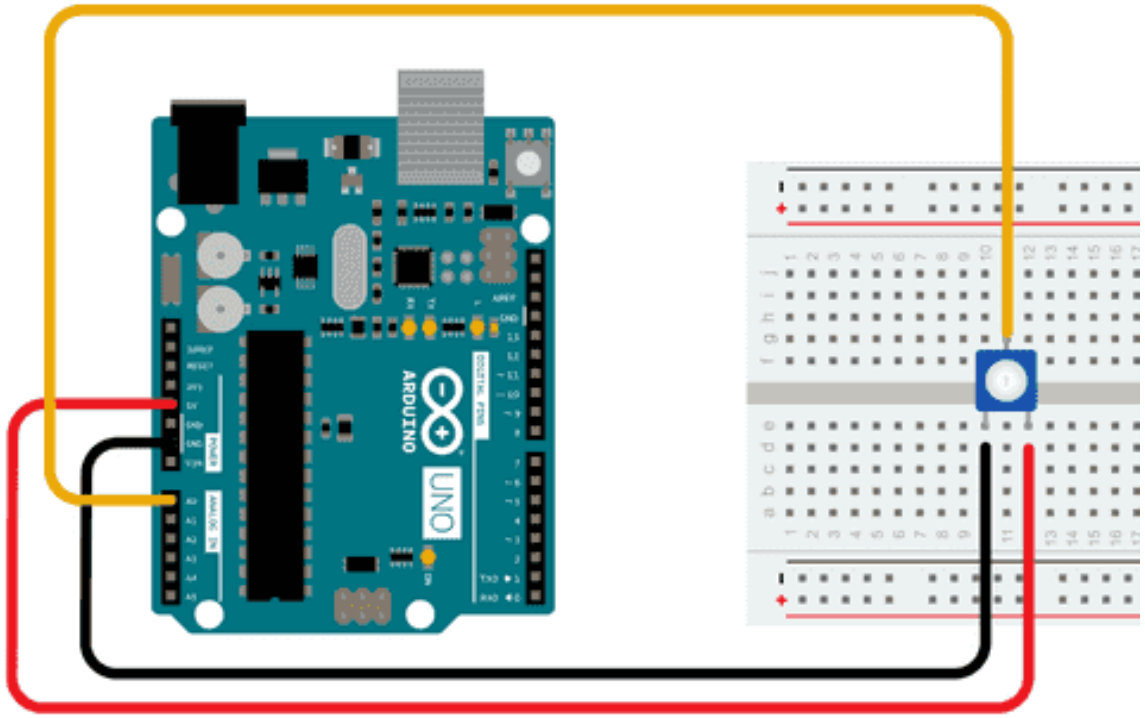
https://docs.arduino.cc/built-in-examples/

- ## Leer una señal analógica

Un potenciómetro es una resistencia variable, nos permitira poner la entrada $A_0$ a una tensión variable entre 0 y 5V. Este ejemplo nos permitirá leer en la pc el valor de tensión sobre la entrada analogica $A_0$

Utilizar un potenciómetro de 10 kΩ



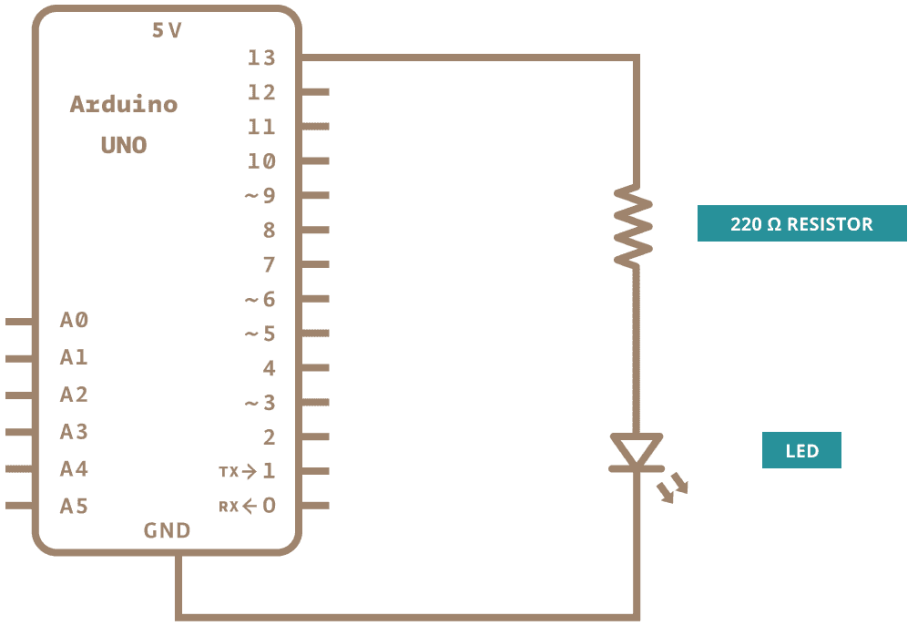
## Leer una señal Analógica

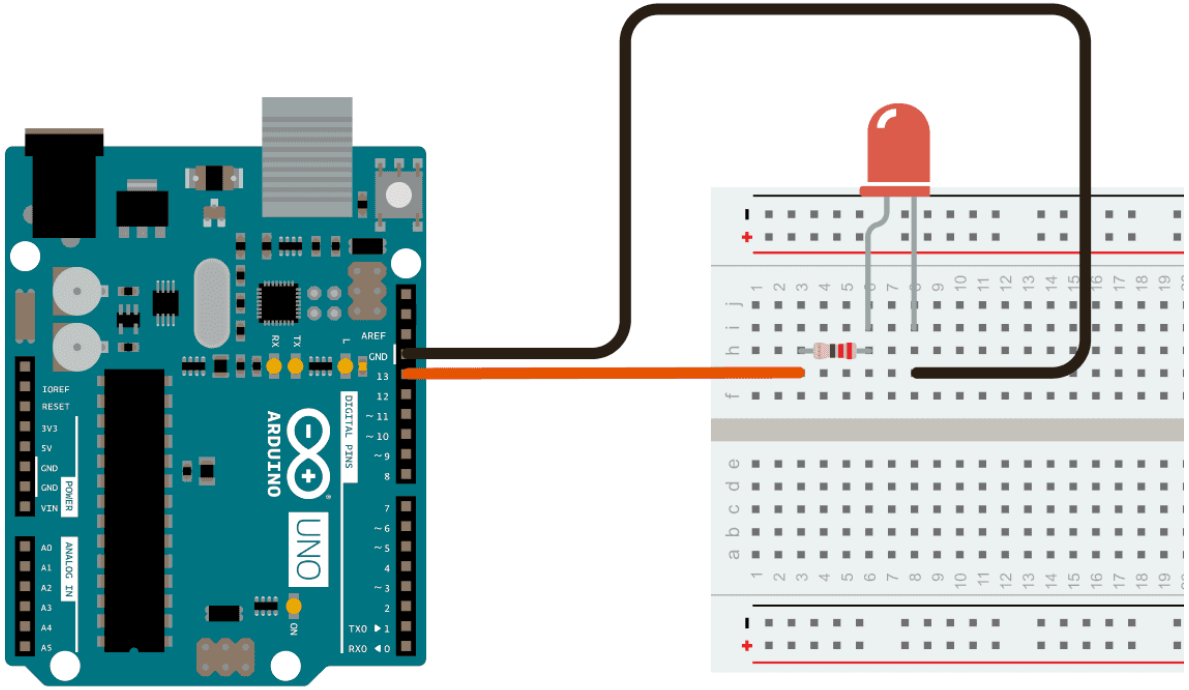
```
  AnalogReadSerial

  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)

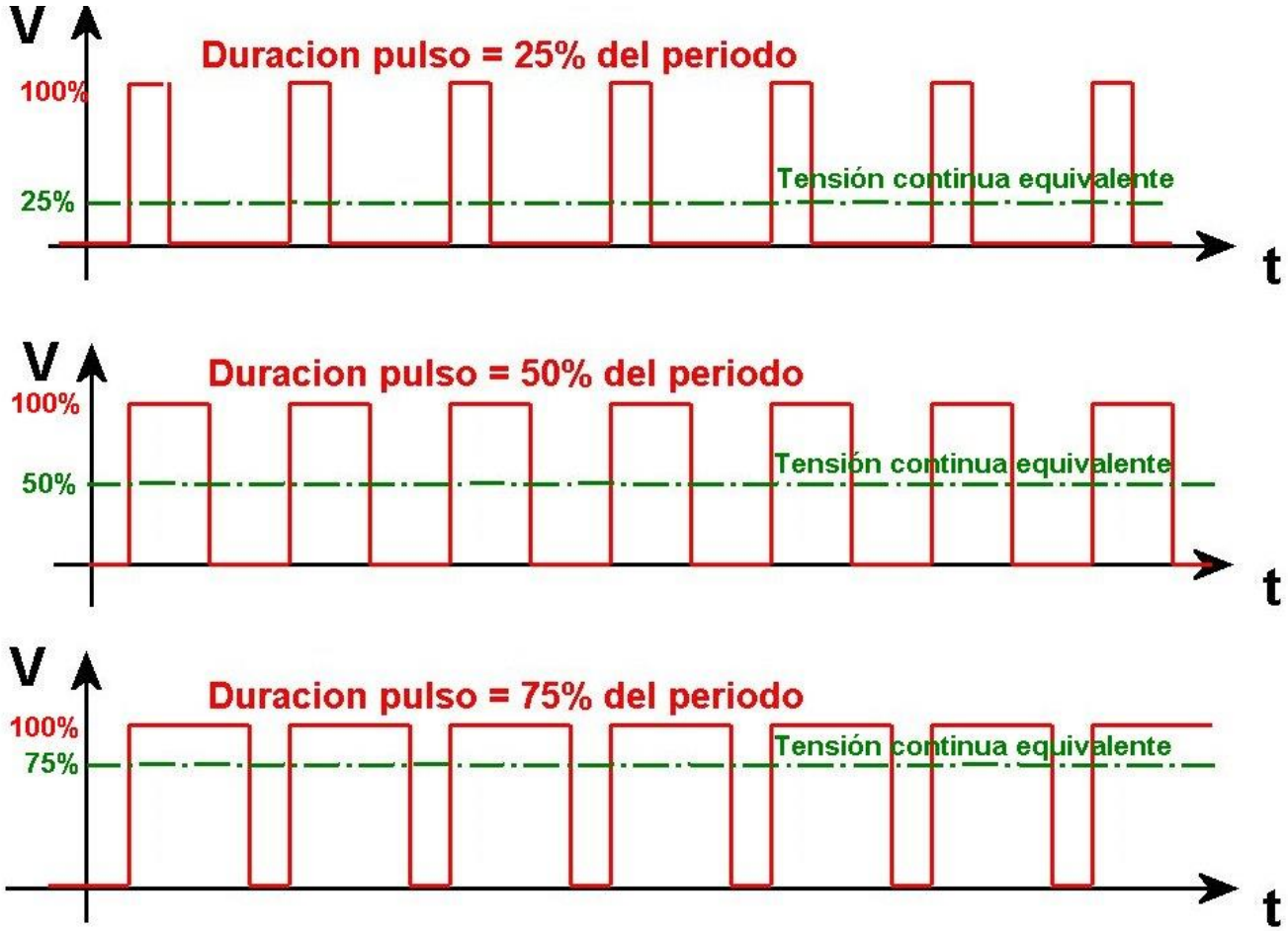


```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// 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
}
```

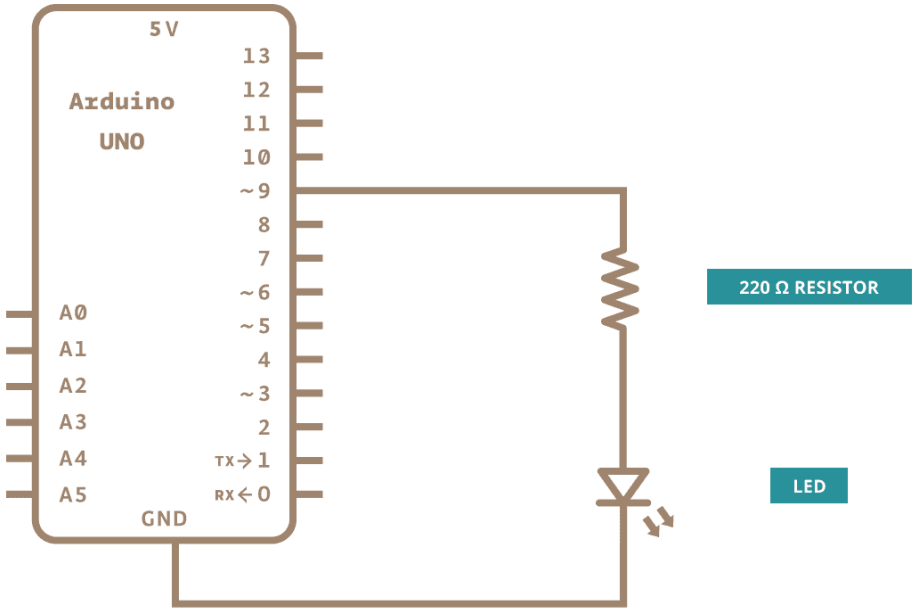
# Control por Ancho de Pulso PWM (pulse width modulation)

Se varia el ciclo de trabajo (Duty Cycle)




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

# 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);
}
```

# Adquirir una 2 canales en función del tiempo. Doscanales_frec_ampli_fase

Resolución en tensión:

- Referencia “Default” 5V Resolución 5V/1024 ≈0.0048V Rango [0, 5] V
- Referencia “Internal” 1.1V Resolución 1.1V/1024 0.0010V Rango [0, 1.1]V

Frecuencia de Adquisición:

- Cambiar el valor de "espera"  útil frecuencias <400Hz
- Cambiar “preescaler” Factor de división del Clock de 16MHz Arduino

Espera

- “0” 451-1000 Hz
- “100” 161-450 Hz
- “300” 74-160 Hz
- “1000” 26-73 Hz

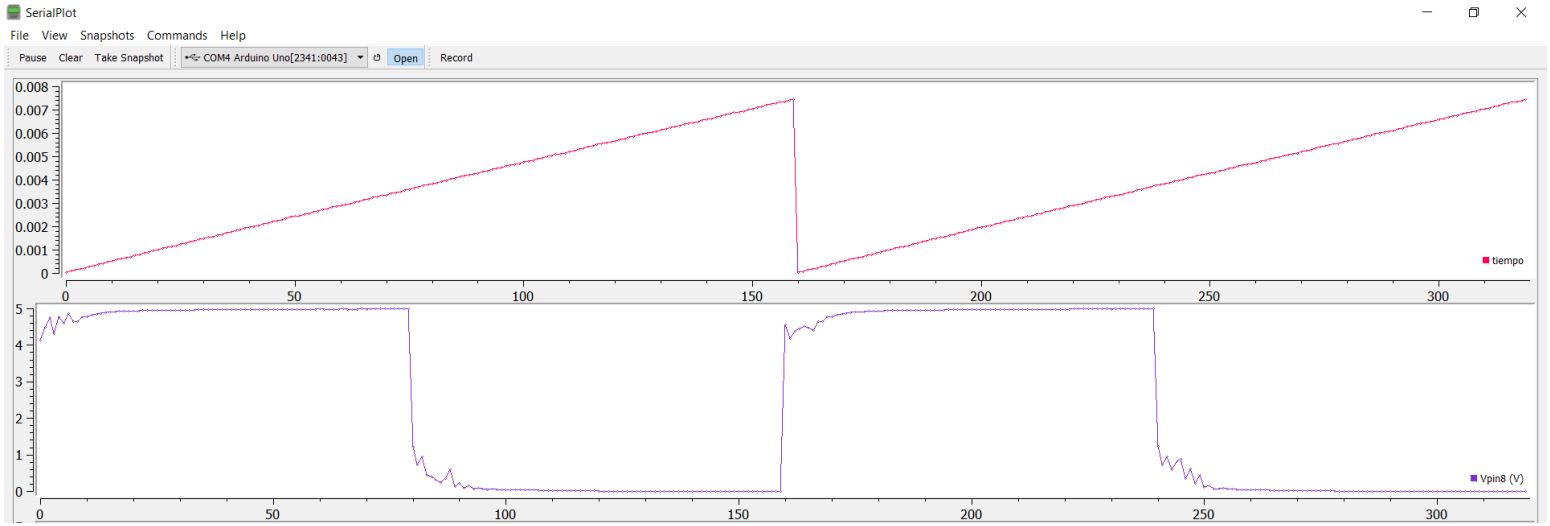
Para frec> 1000 Hz fijar espera=0 y cambiar el prescaler de:

**Ejemplo: 1000 Hz, espera=0 prescaler 04**

Vamos a guardar 130 puntos por canal

## Serial Plot

Nos permitirá ver en tiempo real lo que entra por el puerto serie, reajustando la escala al máximo valor de cada entrada



# Transferir mediciones de Arduino UNO

Serial.print

**Ejecutar Serial Plot**

**Void Loop en ejecución**

**-1) Dejar ejecutándose el IDE de Arduino**
**-2) Chequear los datos adquiridos en el Monitor Serie**
**-3) Ejecutar el Serial Plot (cerrar previamente el Monitor Serie o el Serial Plotter del IDE (si los tuvieran abiertos)**
**-4) Igualar el "Baud Rate" en el "Port" y Setear "Data Format" en ASCII si hiciera falta. Clickear OPEN**
**-5) Visualizar los datos**
**-6) Grabar en archivo en "Record": seleccionar nombre archivo y luego clickear para iniciar y para finalizar sobre RECORD → LISTO! Los datos están en un archivo tipo CSV**

```
 for (int i=0; i<130; i++){
   Serial.print(tiempo[i]/1000000.00000,5); //
tiempo en segundos con 5 dígitos de resolución
   Serial.print(',');
   V1 = canal1[i]*1.1/1023;
   Serial.print(V1,3) ;
   Serial.print(',');
   V2 = canal2[i]*1.1/1023;
   Serial.print(V2,3);
   Serial.print(',');
   Serial.print(Amplitud1,3);
   Serial.print(',');
   Serial.print(Amplitud2,3);
   Serial.print(',');
   Serial.print(frec1,1);
   Serial.print(',');
   Serial.print(Amplitud2/Amplitud1,3);
   Serial.print(',');
   Serial.println(defasaje,1);
  }
 delay(10000);  // espera de 10 s para reiniciar las
mediciones
```

# Actividad

## Utilizar el programa: Doscanales_frec_ampli_fase

- Generar una señal (Senoidal o triangular) en el rango [0, 5] V o [0, 1.1] V en función de la referencia utilizada
- Caracterizarla para distintas frecuencias variando la espera o el preescaler. Iniciando con una frecuencia de 1000Hz
- Probar con 2 señales en simultaneo de distinta amplitud y fase
- Graficar en su programa amigo en modo V(t) y X vs Y