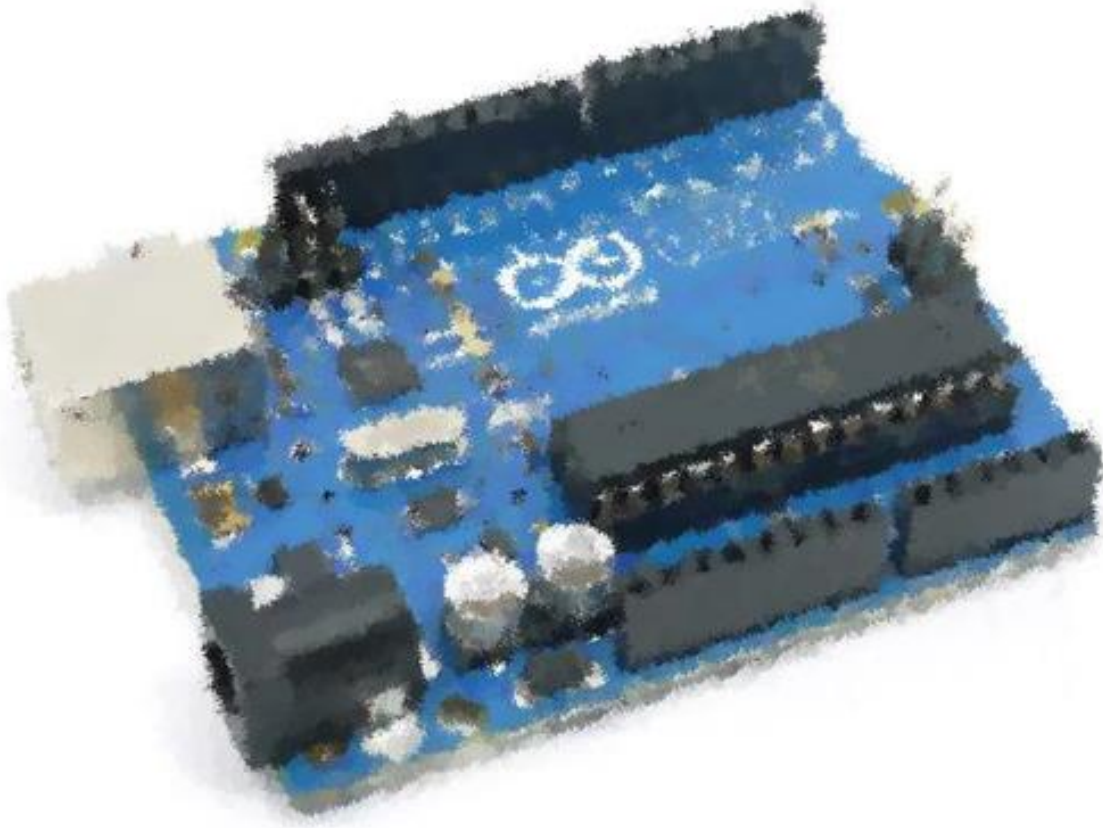


CAPÍTULO

3 nivel
aficionado



Daniel Gallardo García
Profesor de Tecnología
Jerez de la Frontera



Índice



Índice

[Números aleatorios](#)

[Reproducir sonido](#)

[Operadores matemáticos](#)

[Sensores Analógicos: la NTC](#)

[Servomotor 0° - 180°](#)

[Servomotor de rotación continua](#)

[Motor de corriente continua](#)

[Problemas de Arduino con los motores/servomotores](#)

Daniel Gallardo García
Profesor de Tecnología
Jerez de la Frontera



En muchas ocasiones es muy útil utilizar números aleatorios, seleccionados al azar. Comportamientos inesperados, suerte, espontaneidad,... son fruto de emplear números aleatorios. Veamos cómo trabajarlos con Arduino:

La función que genera un número aleatorio es la misma que en Processing, pero con la diferencia de que ahora son números enteros:

```
random(200);           /*asigna a la variable val un valor aleatorio  
                       comprendido entre 0 y 199 */
```

```
random(100, 200);     /*asigna a la variable val un valor aleatorio  
                       comprendido entre 100 y 199 */
```

Ejemplo: Dado trucado



Realiza el siguiente sketch. No hace falta ningún circuito, solo la placa Arduino. Simulará a un dado que saca valores de 1 a 6. Para empezar una nueva tanda de tiradas, basta con apretar el botón de Reset de Arduino para que comience nuevamente el sketch desde el principio:

```
int dado; //será el número obtenido en una tirada

void setup() {
  Serial.begin(9600);
  Serial.println("Empecemos una nueva tanda de tiradas:");
}

void loop() {
  dado = random(1, 7); //asigno a dado un valor aleatorio entre 1 y 6
  Serial.println(dado);
  delay(500);
}
```

¿Qué observas?

¡Siempre es la misma secuencia de números! ¡Esto no es aleatorio!

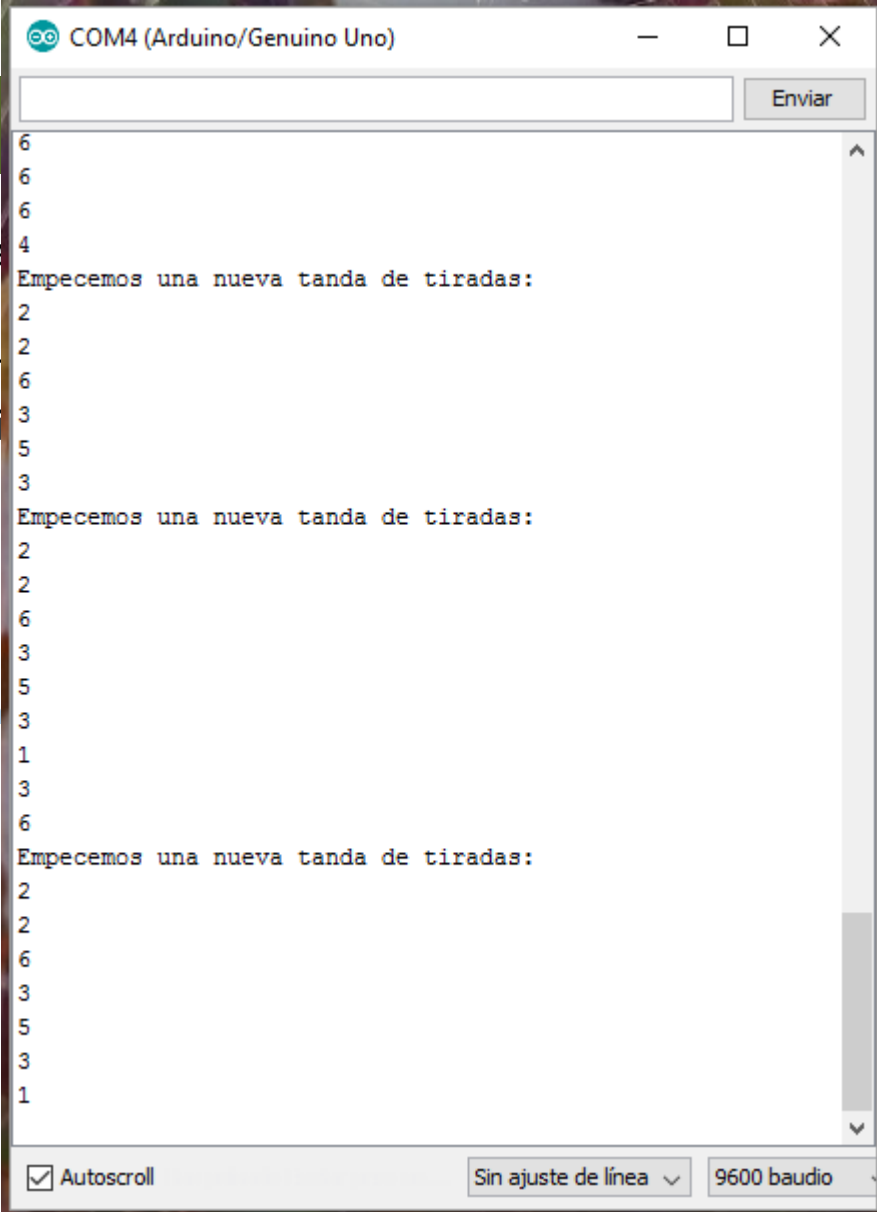
Ejemplo: Dado trucado

Realiza el siguiente sketch. No ha
Simulará a un dado que saca valo
tiradas, basta con apretar el bo
nuevamente el sketch desde el pri

```
int dado; //s  
  
void setup() {  
  Serial.begin(9600);  
  Serial.println("Empecemos una  
}  
  
void loop() {  
  dado = random(1, 7); //a  
  Serial.println(dado);  
  delay(500);  
}
```

¿Qué observas?

¡Siempre es la misma secuencia de números! ¡Esto no es aleatorio!



```
COM4 (Arduino/Genuino Uno)  
Enviar  
6  
6  
6  
4  
Empecemos una nueva tanda de tiradas:  
2  
2  
2  
6  
3  
5  
3  
Empecemos una nueva tanda de tiradas:  
2  
2  
6  
3  
5  
3  
1  
3  
6  
Empecemos una nueva tanda de tiradas:  
2  
2  
6  
3  
5  
3  
1
```

Autoscroll Sin ajuste de línea 9600 baudio





Es posible iniciar la generación de números aleatorios de una manera más impredecible, utilizando las señales recogidas en un pin flotante de entrada analógica, que recoja el ruido de fondo electromagnético (ondas de radio, rayos cósmicos, interferencias electromagnéticas de teléfonos móviles y luces fluorescentes, etc...). En este caso debemos incluir en el setup:

```
randomSeed(analogRead(A0)); /*genera números aleatorios utilizando el dato  
de la lectura del pin de entrada analógica A0 */
```

Ejemplo: Dado mejorado



Utilizaremos `randomSeed()`; para crear números aleatorios que no se repitan, pues la generación de dichos números va a depender del valor de lectura de una entrada digital, que dependerá de las condiciones electromagnéticas ambientales que continuamente están cambiando (lo cual lo convierte de por sí en un valor aleatorio):

```
int dado;                                //será el número obtenido en una tirada

void setup() {
  randomSeed(analogRead(A0));            /*la generación de números aleatorios
  dependerá del valor de lectura de A0, que es aleatorio por el ruido de fondo */
  Serial.begin(9600);
  Serial.println("Empecemos una nueva tanda de tiradas:");
}

void loop() {
  dado = random(1, 7);                    //asigno a dado un valor aleatorio entre 1 y 6
  Serial.println(dado);
  delay(500);
}
```

¡Ahora no se repite la misma secuencia de números aleatorios!

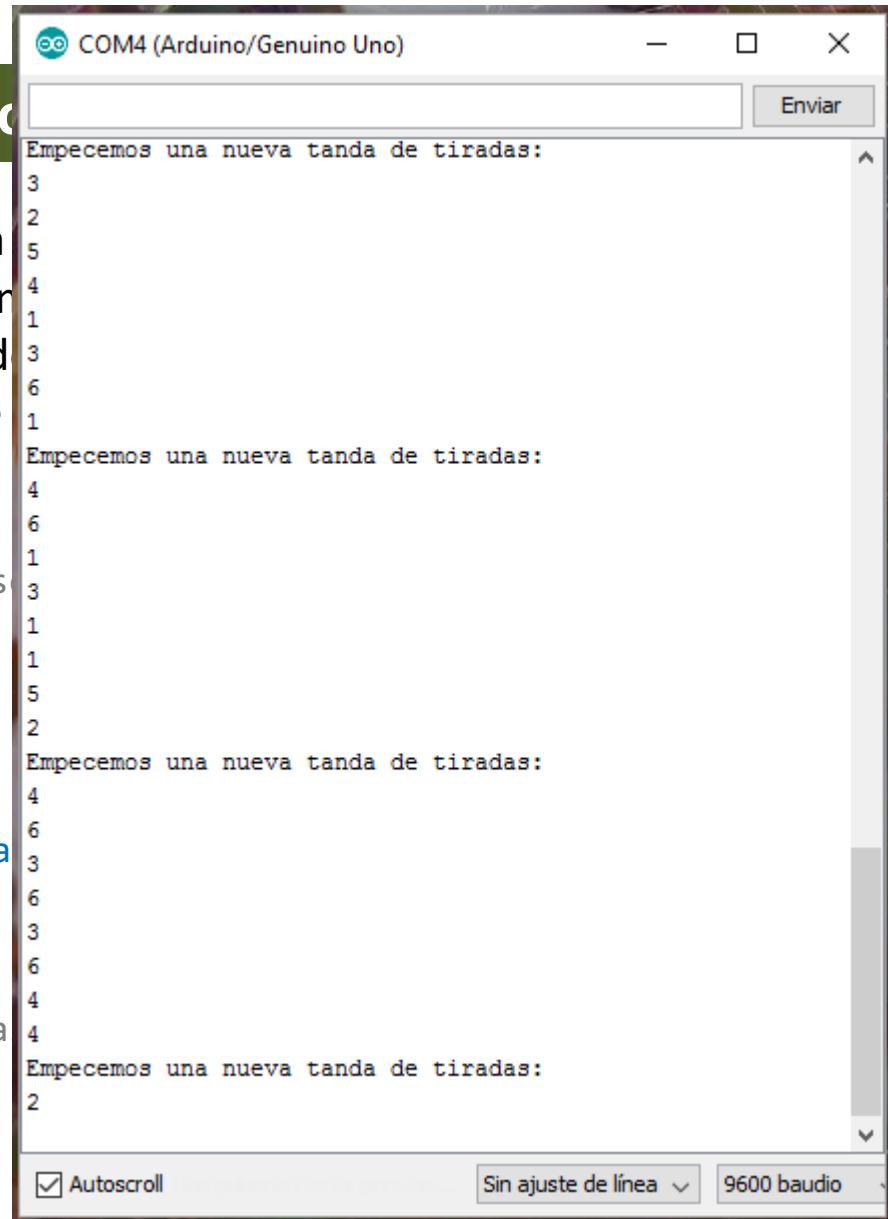
Ejemplo: Dado mejorado

Utilizaremos `randomSeed()`; para pues la generación de dichos números de entrada digital, que depende de factores ambientales que continuamente cambian (dependen de un valor aleatorio):

```
int dado; //s

void setup() {
  randomSeed(analogRead(A0));
  dependerá del valor de lectura
  Serial.begin(9600);
  Serial.println("Empecemos una
}

void loop() {
  dado = random(1, 7); //a
  Serial.println(dado);
  delay(500);
}
```



```
COM4 (Arduino/Genuino Uno)
Enviar
Empecemos una nueva tanda de tiradas:
3
2
5
4
1
3
6
1
Empecemos una nueva tanda de tiradas:
4
6
1
3
1
5
2
Empecemos una nueva tanda de tiradas:
4
6
3
6
3
4
4
2
Autoscroll Sin ajuste de línea 9600 baudio
```

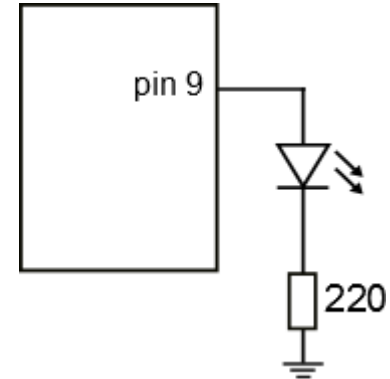
¡Ahora no se repite la misma secuencia de números aleatorios!



Ejemplo: Candela en el portal de Belén



Utilizaremos un LED rojo o naranja para imitar una candela. Para ello haremos que el brillo y su duración vaya cambiando aleatoriamente:



```
int brillo;
int tiempo;

void setup() {
  randomSeed(analogRead(A3)); //generaré números aleatorios con A3
  pinMode(9, OUTPUT); //debe de estar conectado a una salida analógica
}

void loop() {
  brillo = random(50, 256); //el brillo será aleatorio
  analogWrite(9, brillo);
  tiempo = random(50, 151); //el tiempo también será aleatorio
  delay(tiempo);
}
```

Reproducir sonido

`tone(4, 1000);`



Para reproducir sonido utilizaremos como dispositivo de salida un **zumbador** o un **altavoz**, cuyos símbolos son:



Las funciones para reproducir sonido son:

```
tone(6, 1350);
```

```
/*el altavoz conectado al pin 6 emitirá un sonido de 1350 Hz de duración indeterminada. Tras un tone se suele poner un delay() que permita escuchar esa frecuencia haciendo una pausa en el programa*/
```

```
noTone(6);
```

```
//el altavoz conectado al pin 6 dejará de sonar
```


```
tone(9, 4500, 200);
```

```
/*el altavoz conectado al pin 9 emitirá un sonido de 4500 Hz durante un tiempo de 200 milisegundos y luego se detendrá */
```

Reproducir sonido_2



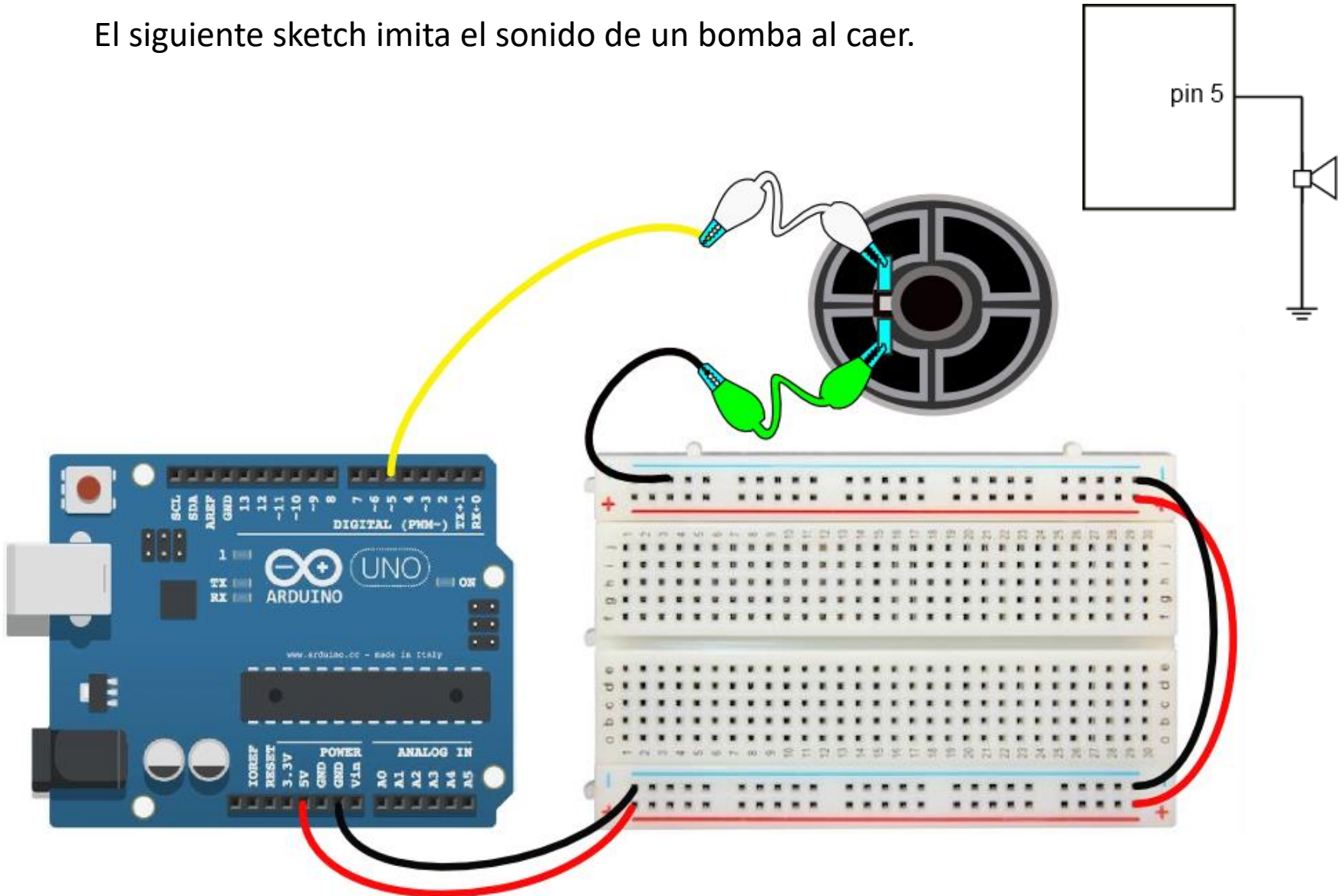
Puede ser muy útil saber qué frecuencia, en hertzios, corresponde a cada nota musical por si queremos que Arduino nos recite alguna canción (se recuerda que para expresar los decimales se emplea el punto y no la coma). Es bueno saber que tampoco se va a notar mucho si redondeo estas frecuencias a números enteros.

	Frecuencias (Hz) de las notas musicales							
	ESCALA							
NOTA	1	2	3	4	5	6	7	8
Do	65.406	130.813	261.626	523.251	1046.502	2093.005	4186.009	8372.018
Do#	69.296	138.591	277.183	554.365	1108.731	2217.461	4434.922	8869.844
Re	73.416	146.832	293.665	587.33	1174.659	2349.318	4698.636	9397.273
Re#	77.782	155.563	311.127	622.254	1244.508	2489.016	4978.032	9956.063
Mi	82.407	164.814	329.628	659.255	1318.51	2637.02	5274.041	10548.082
Fa	87.307	174.614	349.228	698.456	1396.913	2793.826	5587.652	11175.303
Fa#	92.499	184.997	369.994	739.989	1479.982	2959.955	5919.911	11839.822
Sol	97.999	195.998	391.995	783.991	1567.982	3135.963	6271.927	12543.854
Sol#	103.826	207.652	415.305	830.609	1661.219	3322.438	6644.875	13289.75
La	110	220	440	880	1760	3520	7040	14080
La#	116.541	233.082	466.164	932.328	1864.655	3729.31	7458.62	14917.24
Si	123.471	246.942	493.883	987.767	1975.533	3951.066	7902.133	15804.266

Ejemplo: ¡La bomba!



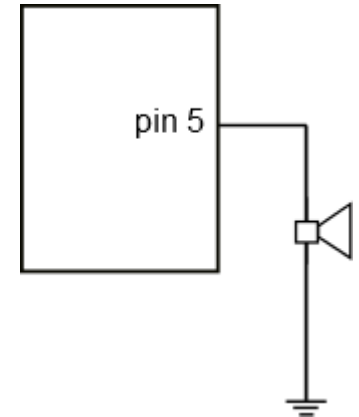
El siguiente sketch imita el sonido de un bomba al caer.



Ejemplo: ¡La bomba!



El siguiente sketch imita el sonido de un bomba al caer.



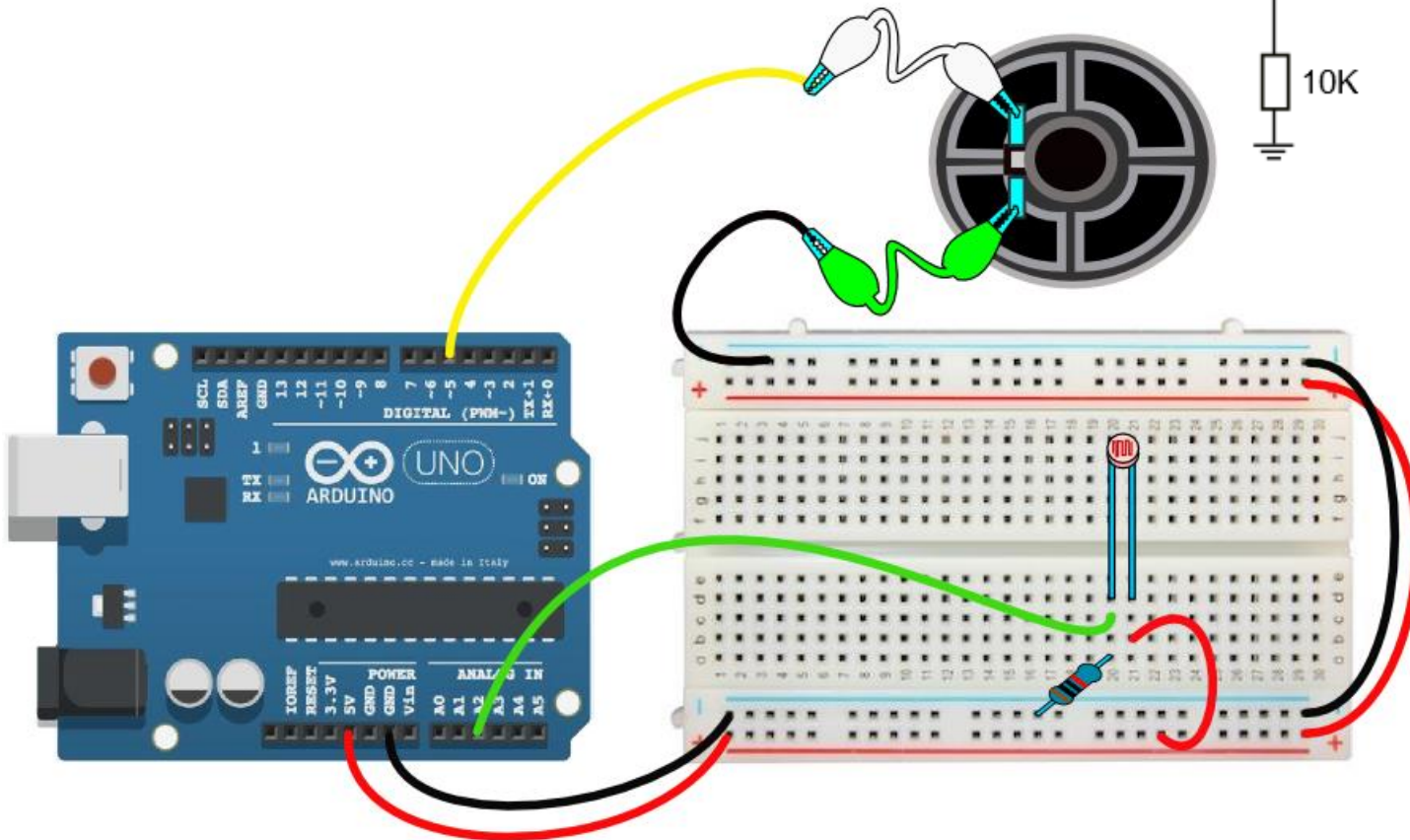
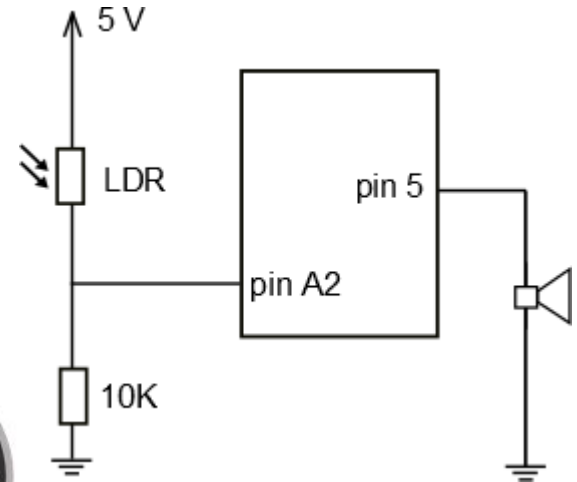
```
void setup() {  
  pinMode(5, OUTPUT);  
}  
  
void loop() {  
  for(int i=5000; i>50; i--) {  
    tone(5, i);  
    delay(3);  
  }  
}
```

```
//la frecuencia del sonido irá disminuyendo  
//cada nota tendrá una duración de 3 ms  
//en total habrán sonado 4949 notas distintas
```

Ejemplo: Instrumento musical invisible (Theremin)



Crearemos un instrumento musical que se controla con la posición de la mano (más o menos cerca de una LDR, que será empleada como sensor de proximidad, aprovechando que cuanto más cerca esté la mano, menos luz le llegará).



Ejemplo: Instrumento musical invisible (Theremin)

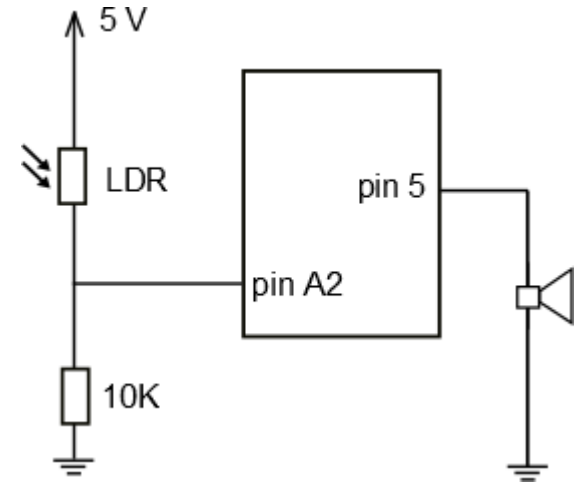


Crearemos un instrumento musical que se controla con la posición de la mano (más o menos cerca de una LDR, que será empleada como sensor de proximidad, aprovechando que cuanto más cerca esté la mano, menos luz le llegará).

```
int luz;           //será el valor de entrada de la LDR
int frecuencia;   //será la frecuencia del altavoz

void setup() {
  pinMode(5, OUTPUT);
  Serial.begin(9600); //interesa ver los valores de la LDR
}

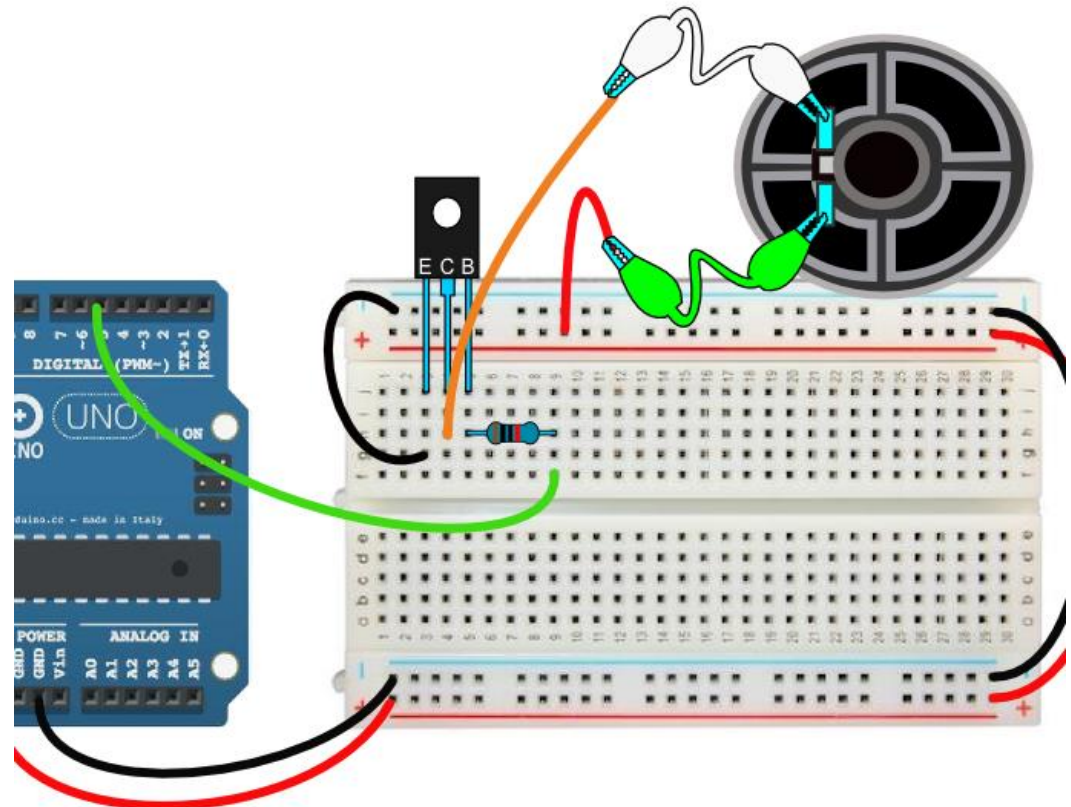
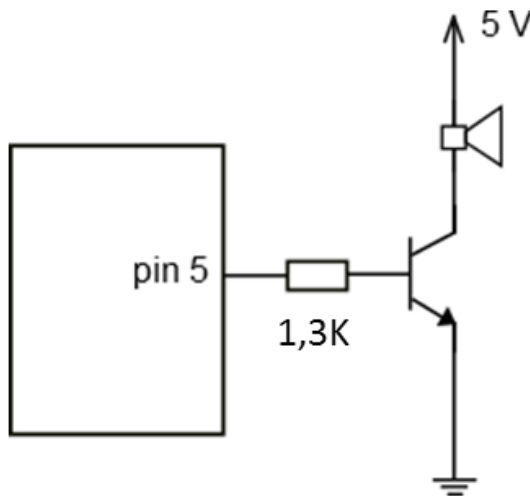
void loop(){
  luz = analogRead(A2);
  Serial.print("lectura de entrada: ");
  Serial.print(luz);
  Serial.print('\t');
  frecuencia = map(luz, 20, 500, 3000, 20); //deberíamos afinar más el rango inicial
  frecuencia = constrain(frecuencia, 20, 3000); //obligo a que no salga de esos valores
  Serial.print("frecuencia sonido: ");
  Serial.println(frecuencia);
  tone(5, frecuencia, 200);
  delay(250); //doy tiempo a que suene la nota y a una pequeña pausa
}
```



Aumentar el volumen del altavoz



La intensidad de corriente máxima que Arduino es capaz de generar por un pin de salida es de 40 mA. Sin embargo, por el pin de 5 V la intensidad máxima es de 300 mA (7,5 veces más). Si utilizo un transistor funcionando como interruptor, podemos hacer que nuestro altavoz suene con bastante más volumen:



Operadores matemáticos



Los operadores aritméticos que se ya hemos utilizado en los sketches anteriores han sido:

asignación	=
suma	+
resta	-
multiplicación	*
división	/

Falta uno: el operador módulo, que retorna el resto de una división (ya lo vimos en Processing):

módulo	%
--------	---

Veamos cómo funcionan:

```
float x;  
int y;  
int z;
```

```
x = 7 / 2;           //x tomará el valor de 3.5 puesto que está declarado como float
```

```
y = 7 / 2;           //y tomará el valor de 3 puesto que no puede tener decimales
```

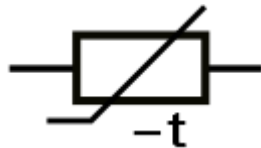
```
z = 7 % 2;           //z tomará el valor 1 puesto que es el resto de dividir 7 / 2
```

Sensores Analógicos: La NTC

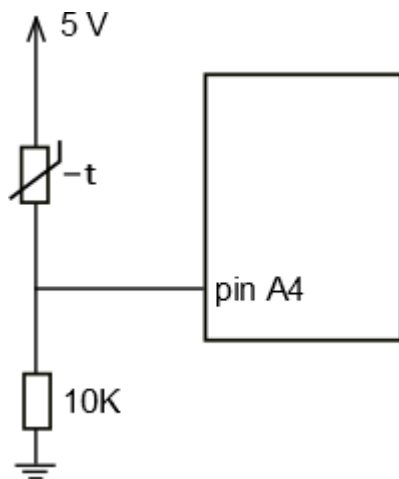


Una NTC es una resistencia variable con la temperatura: cuanto mayor sea la temperatura, menos ohmios tendrá. Podemos utilizar una NTC para utilizarlo como sensor de temperatura.

Su símbolo es el siguiente:



Podemos conseguir una entrada analógica de la siguiente manera:



T^a alta \rightarrow `analogRead(A4)` = valores altos (~ 500)

...

T^a moderada \rightarrow `analogRead(A4)` = valores medios (~ 300)

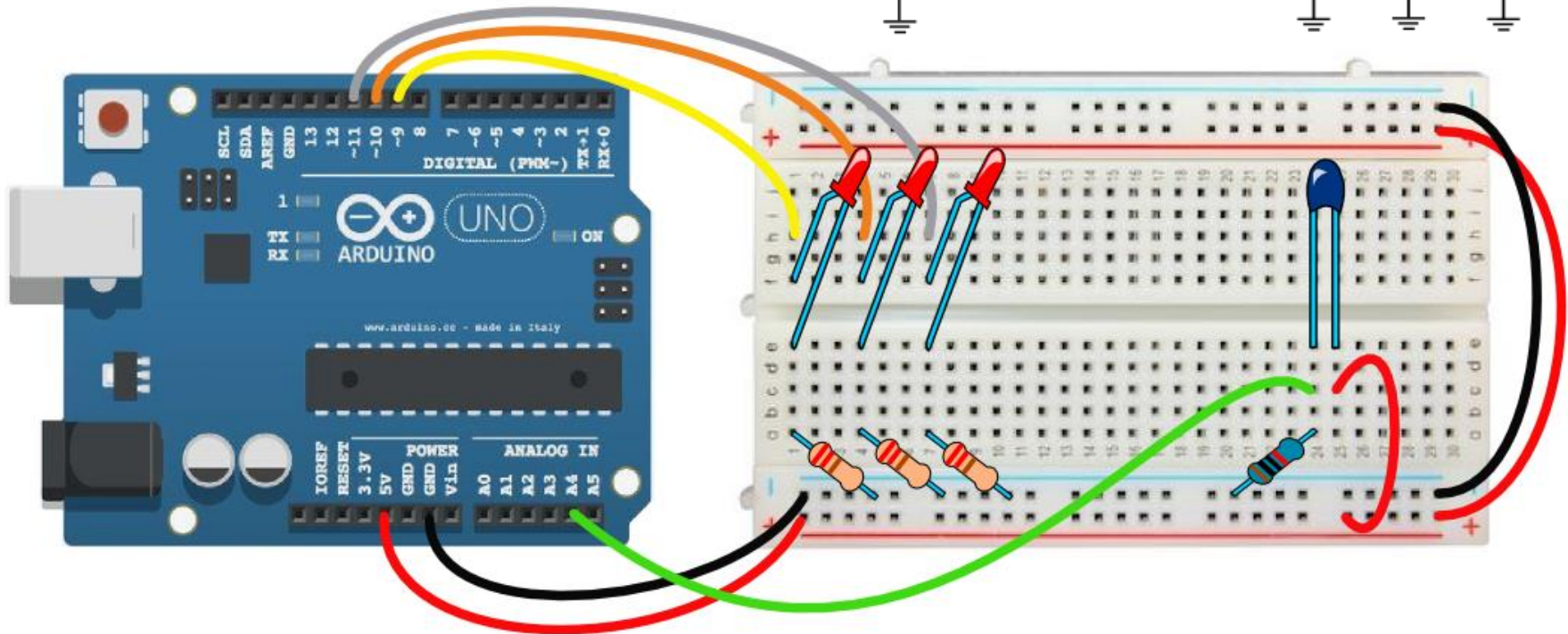
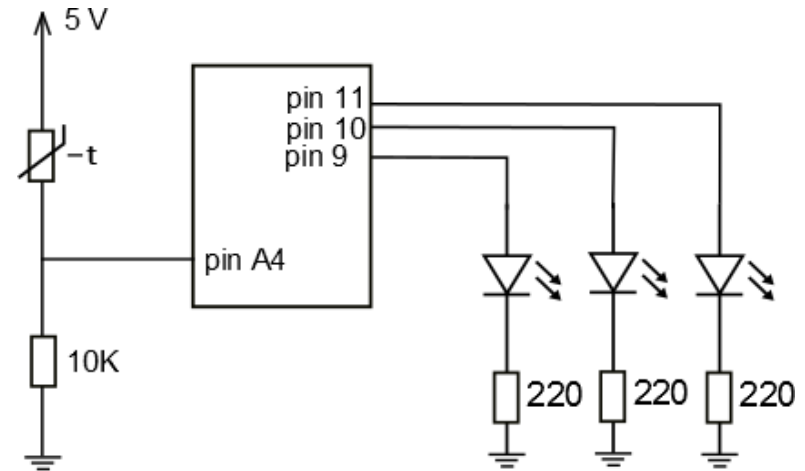
...

T^a baja \rightarrow `analogRead(A4)` = valores bajos (~ 100)

Ejemplo: Termómetro de LEDs



Construiremos un termómetro a través de 3 LEDs alineados, y a medida que la temperatura aumente, la luz irá subiendo por los distintos LEDs de manera gradual.



Ejemplo: Termómetro de LEDs



```
void setup() {  
  for(int i=9; i<12; i++) pinMode(i,OUTPUT);  
  Serial.begin(9600);  
}
```

```
void loop() {  
  Serial.println(analogRead(A4));  
  
  int x = map(lectura, 280, 400, 0, 30);  
  x = constrain(x, 0, 30);  
  int entero = x / 10;  
  int resto = x % 10;  
  int brillo = map(resto, 0, 9, 0, 255);  
  if(entero < 1) {  
    analogWrite(9, brillo);  
    digitalWrite(10, LOW);  
    digitalWrite(11, LOW);  
  }  
  else if(entero < 2) {  
    digitalWrite(9, HIGH);  
    analogWrite(10, brillo);  
    digitalWrite(11, LOW);  
  }  
}
```

/*me interesa conocer los valores máx y min para hacer bien el programa */
//mapeo a 30 valores posibles

//solo podrá haber 4 casos: <1, <2, <3, =3
//podrá haber 10 casos: 0, 1, 2, ..., 9
//valores medios para el brillo del LED

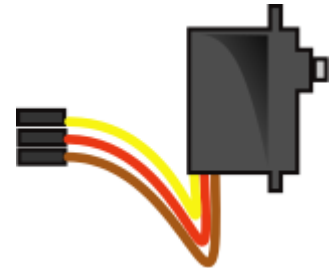
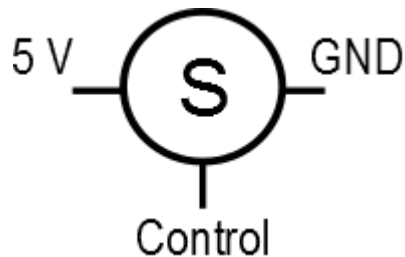
```
    else if(entero < 3) {  
      digitalWrite(9, HIGH);  
      digitalWrite(10, HIGH);  
      analogWrite(11, brillo);  
    }  
    else {  
      digitalWrite(9, HIGH);  
      digitalWrite(10, HIGH);  
      digitalWrite(11, HIGH);  
    }  
    delay(200);  
  }
```

Servomotor 0° - 180°



Un servomotor es un motor que se caracteriza por su **precisión**, pues puede situarse en cualquier posición dentro de un rango de giro, normalmente **de 0° a 180°**. Así pues no son motores pensados para hacer mover un vehículo que recorra cierta distancia, sino para movimientos de precisión como pudiera ser el de un brazo robot, cuyo margen de maniobra no exceda dicho rango de giro.

Su símbolo es el siguiente:



Son tres los cables de que dispone el servomotor, y debemos conectarlos de manera correcta:

Cable **rojo**: se conectará a **5 V**.

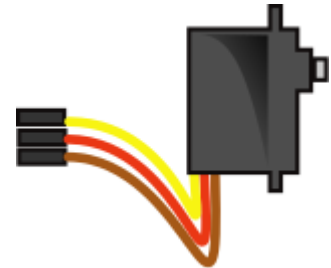
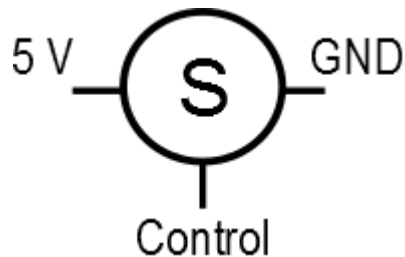
Cable **marrón** o **negro**: se conectará a tierra (**GND**).

Cable **amarillo**, **naranja** o **blanco**: se conectará al **pin** de control (del 0 al 13).

Servomotor 0° - 180°



A la hora de trabajar con motores o servomotores, debido a su alto consumo de corriente, es muy deseable utilizar una alimentación exterior, ya sea una fuente de alimentación o pilas, pues por el puerto de un PC no sale excesiva corriente y provoca malfuncionamiento (como “temblar” los servomotores).



Son tres los cables de que dispone el servomotor, y debemos conectarlos de manera correcta:

Cable **rojo**: se conectará a **5 V**.

Cable **marrón** o **negro**: se conectará a tierra (**GND**).

Cable **amarillo**, **naranja** o **blanco**: se conectará al **pin** de control (del 0 al 13).

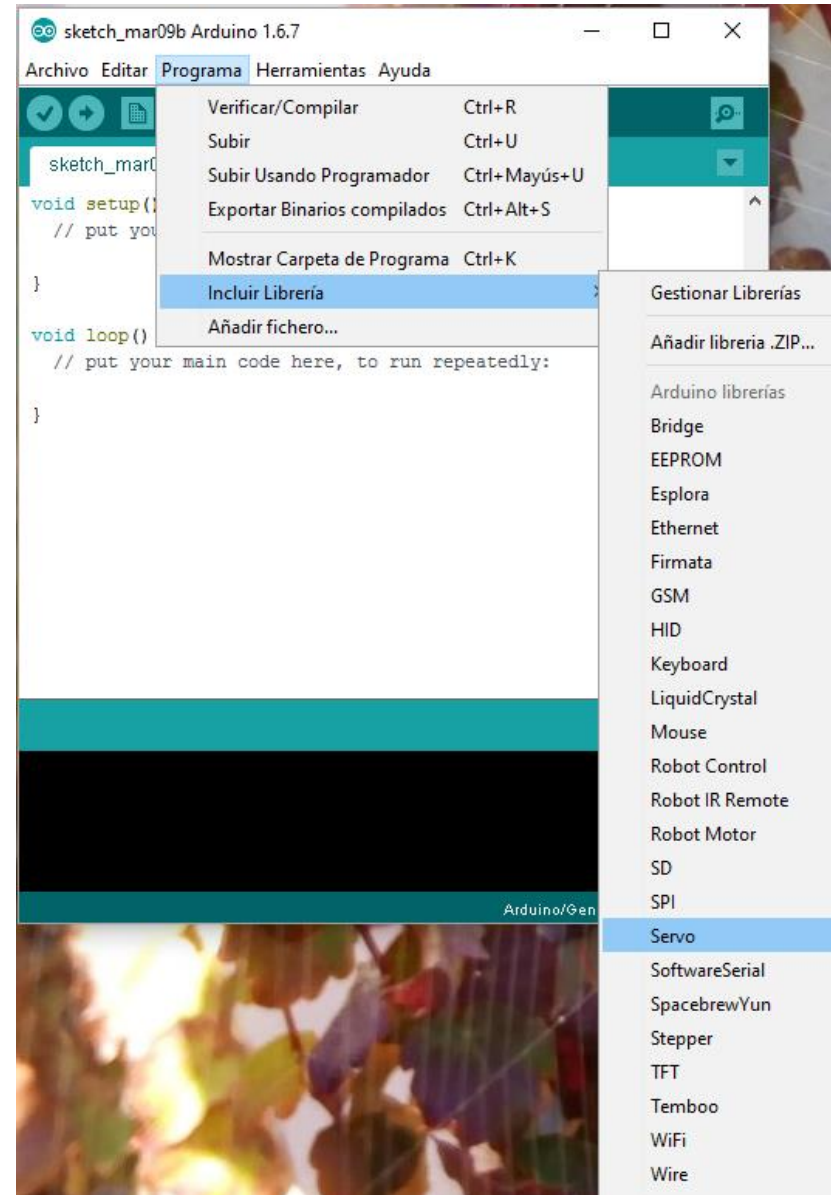
Servomotor 0° - 180°_2

#include <Servo.h>



Para el control de los servomotores Arduino posee una **librería** específica. Una librería es una colección de funciones que están especialmente creadas para facilitar el manejo de ciertos dispositivos, y que no son cargadas por defecto a Arduino para ahorrar espacio en su memoria.

En primer lugar debemos **incluir la librería:**
Programa / Incluir Librería / Servo





Luego debemos **declarar**, como si de una variable se tratara, **nuestro servomotor**, asignándole un nombre. Y lo hacemos con el siguiente comando:

```
Servo nombreServo;           //llamaré a mi servo nombreServo
```

Dentro del void setup() debemos **configurar el pin de salida** que Arduino va a utilizar para controlar el servomotor. Para ello se utiliza la función `attach(pin)`:

```
nombreServo.attach(10);      //será controlado por el pin 10
```

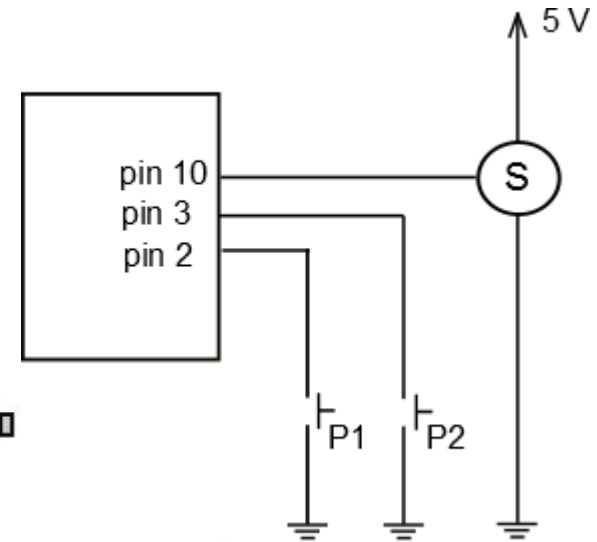
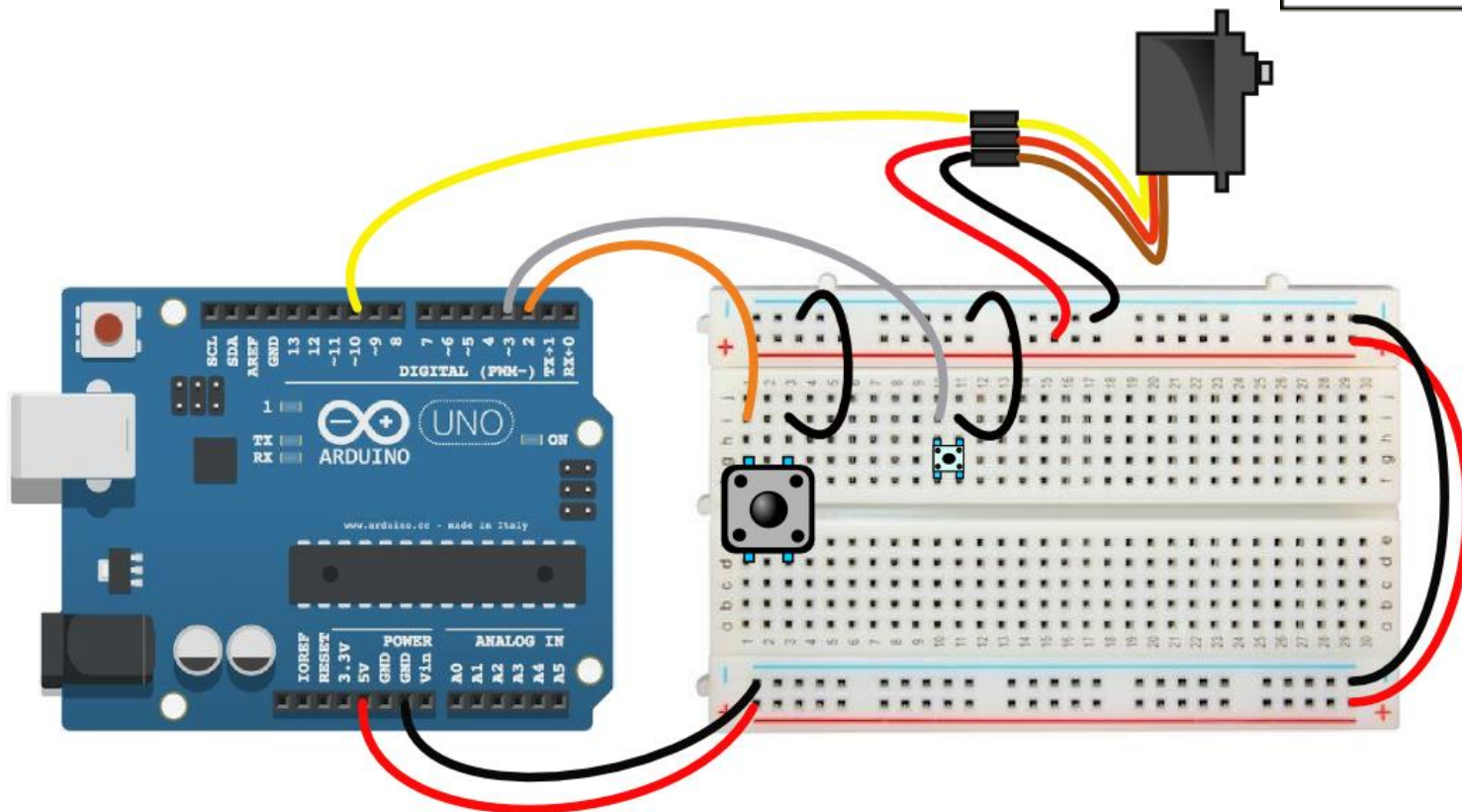
Para **controlar el servomotor**, utilizaremos la función `write(ángulo)`:

```
nombreServo.write(45);      //sitúa el eje del servo en la posición de 45°
```


Ejemplo: Girando voy, girando vengo



Cada vez que pulsemos el pulsador P1, el servomotor girará 10° . Cuando pulsemos el pulsador P2, volverá a su posición original.



Ejemplo: Girando voy, girando vengo

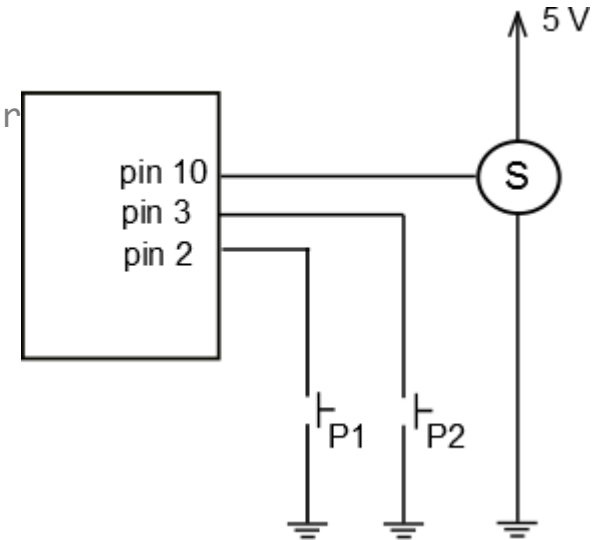


```
#include <Servo.h>
```

```
Servo girador; //declaro nuestro servo, de nombre: girador
int posInicial = 40;
int incremento = 10;
int posActual = posInicial; //al comienzo estará en la
//posición inicial
```

```
void setup() {
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  girador.attach(10); //servo conectado al pin 10
  girador.write(posInicial); //inicialmente a 40°
}
```

```
void loop() {
  if(!digitalRead(2)) { //cuando apretemos P1...
    posActual += incremento; //la posición aumentará en 10°
    girador.write(posActual);
    delay(1000); //hacemos una pausa para que le de tiempo a moverse
  }
  if(!digitalRead(3)) { //cuando apretemos P2...
    girador.write(posInicial); //volverá a la posición inicial
    posActual = posInicial; //y reinicio la posición actual a 40°
    delay(1000);
  }
}
```

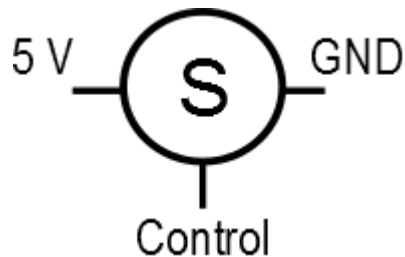


Servomotor de rotación continua



A diferencia del anterior servomotor, éste no tiene limitación en los ángulos de giro, con lo cual es ideal para utilizarlos para vehículos, pudiéndose además controlar la velocidad de giro.

Su símbolo es el siguiente:



Al igual que el servomotor anterior, son tres los cables de que dispone:

Cable **rojo**: se conectará a **5 V**.

Cable **marrón** o **negro**: se conectará a tierra (**GND**).

Cable **amarillo**, **naranja** o **blanco**: se conectará al **pin** de control (del 0 al 13).

Servomotor de rotación continua_2 #include <Servo.h>



El control de este tipo de servomotores utiliza las mismas funciones, pero tienen un comportamiento diferente:

En primer lugar debemos **incluir la librería**: #include <Servo.h>
Programa / Incluir Librería / Servo

Luego debemos **declarar nuestro servomotor**, asignándole un nombre:

```
Servo nombreServo;           //llamaré a mi servo "nombreServo"
```

Dentro del void setup() debemos **configurar el pin de salida** que Arduino va a utilizar para controlar el servomotor:

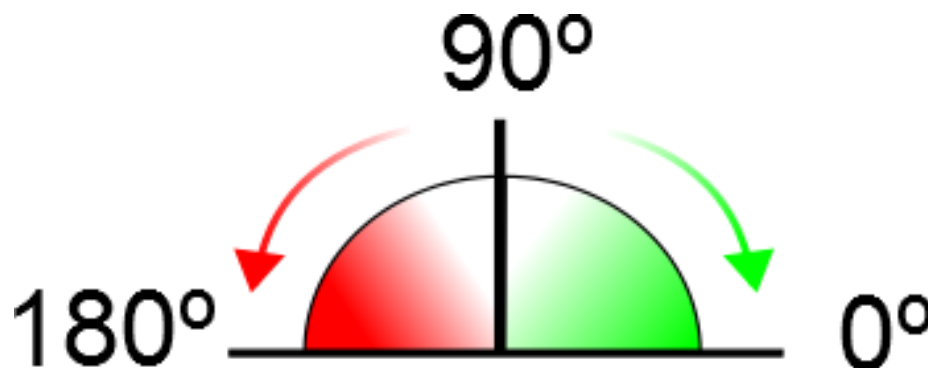
```
nombreServo.attach(10);      //será controlado por el pin 10
```

Servomotor de rotación continua_3



Para controlar el servomotor, utilizaremos la función `write(ángulo)`, pero de la siguiente manera:

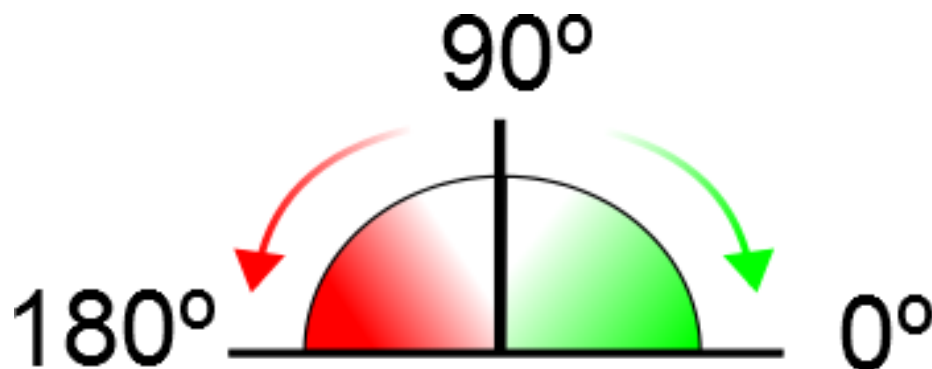
```
nombreServo.write(90);      /*90 hace que no gire: que se detenga. Si no es  
                             así, habrá que calibrarlo con un destornillador */  
nombreServo.write(0);      //velocidad máxima en sentido horario  
  
nombreServo.write(135);    //media velocidad en sentido antihorario
```



Servomotor de rotación continua_3



Es muy frecuente que un valor de 90° no se corresponda al estado de reposo del servomotor de rotación continua. Hay modelos de servomotores de rotación continua que disponen de un calibrador (un tornillito), que debemos ajustar hasta que quede en reposo. A los que no disponen de tal tornillo, habrá que calibrarlo mediante software: por ejemplo con un programa que vaya mostrando mediante el Monitor Serie los valores del argumento de la función `write()` y anotar en cuál se queda quieto (que podría ser, por ejemplo, a un ángulo de 92°).

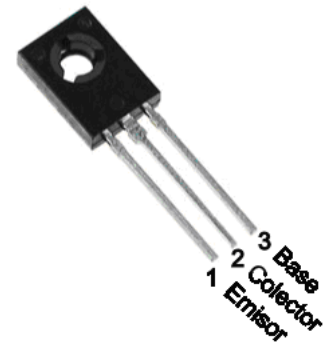
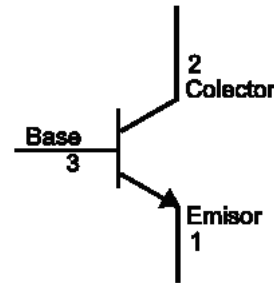
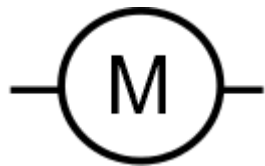


Motor de corriente continua

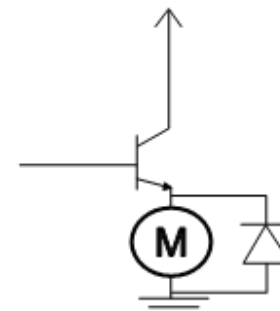
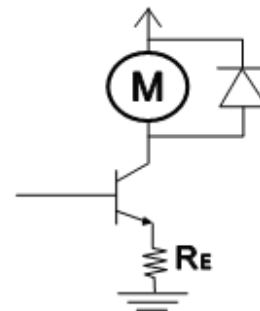
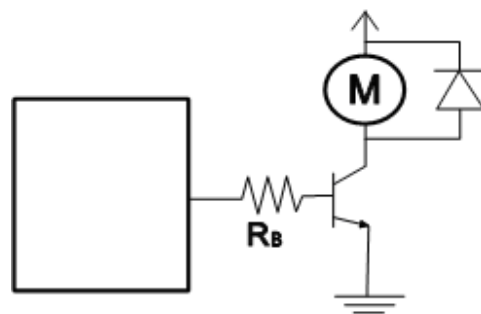


Son los motores más simples: cuando circula una corriente suficiente entre sus bornes, éste girará en un sentido. Para que el motor gire en el sentido contrario, la corriente debe circular en el sentido opuesto al caso anterior.

Su símbolo es el siguiente:



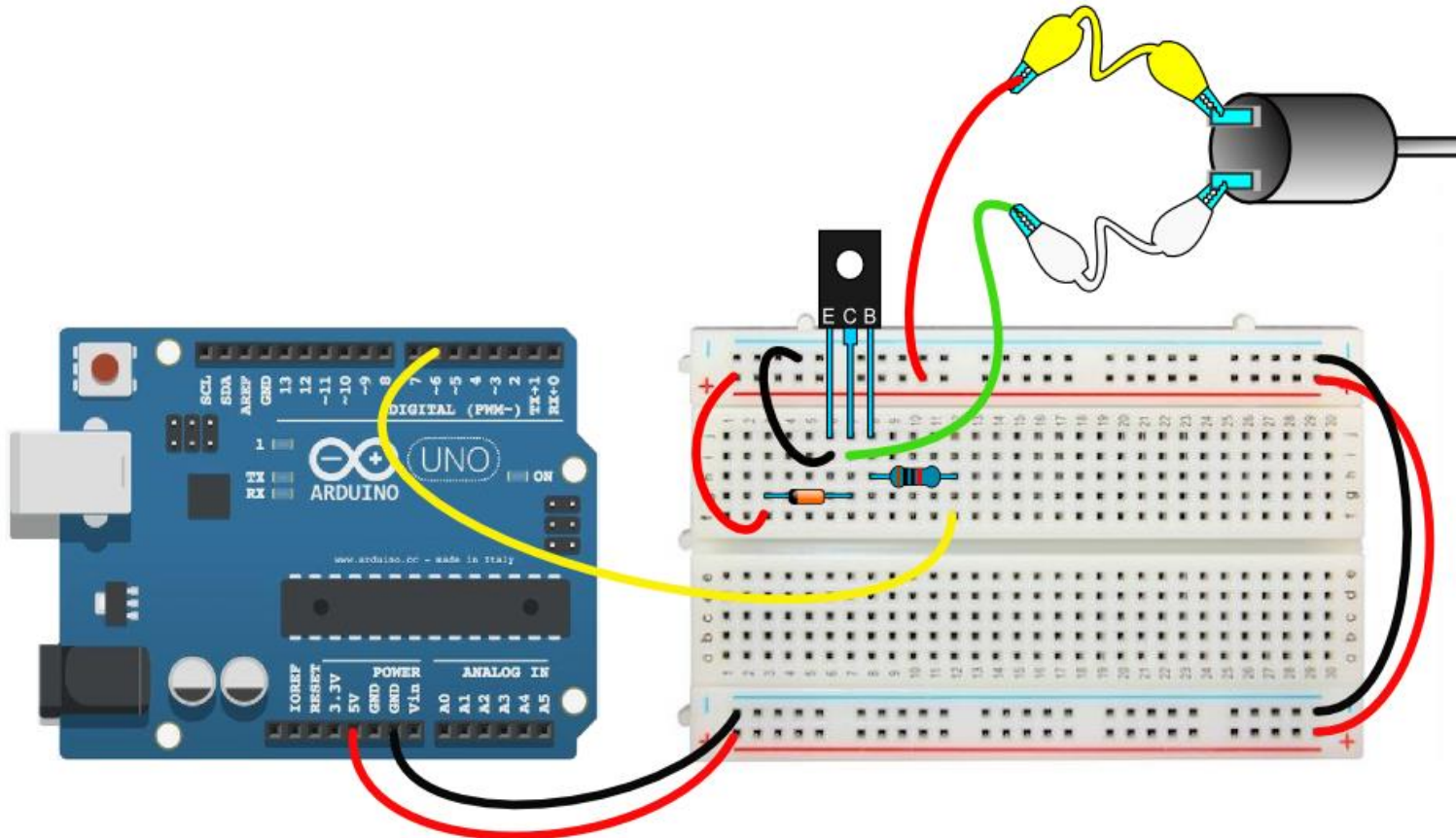
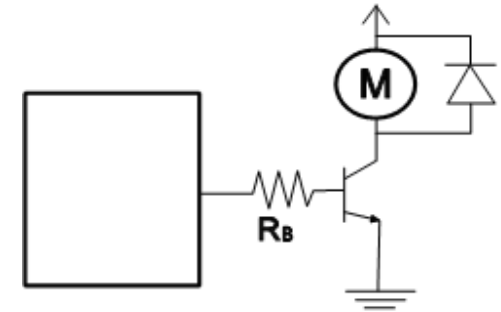
Si solo necesitamos que **gire en un único sentido**, podemos controlarlo con ayuda de un **transistor** (BD135 por ejemplo), una resistencia (1,3K) y un diodo, para realizar alguno de estos montajes (aunque es recomendable la primera opción):



Motor de corriente continua_2



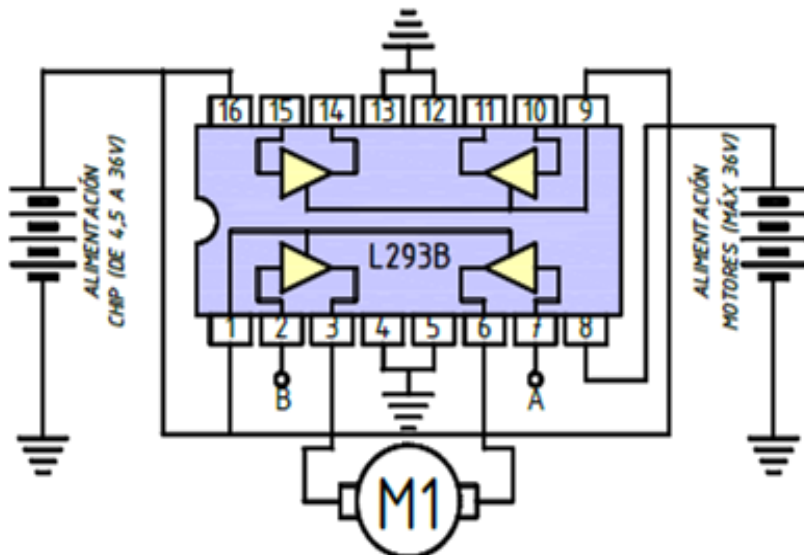
El diodo se utiliza para evitar las corrientes inducidas por las bobinas del motor. Con este montaje podemos **controlar** tanto el **encendido** como el **apagado** del motor, así como su **velocidad** (utilizando un PWM como salida), pero siempre **girando en el mismo sentido**.



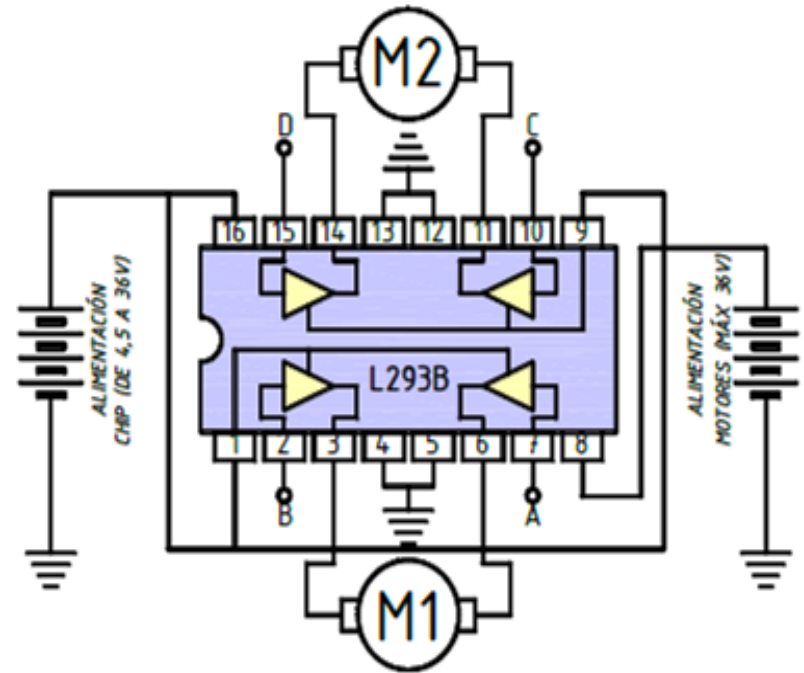
Motor de corriente continua_3



Si lo que necesitamos es que pueda girar en ambos sentidos, debemos ayudarnos del **circuito integrado L293D**, el cual es capaz de controlar hasta dos motores de corriente continua simultáneamente:

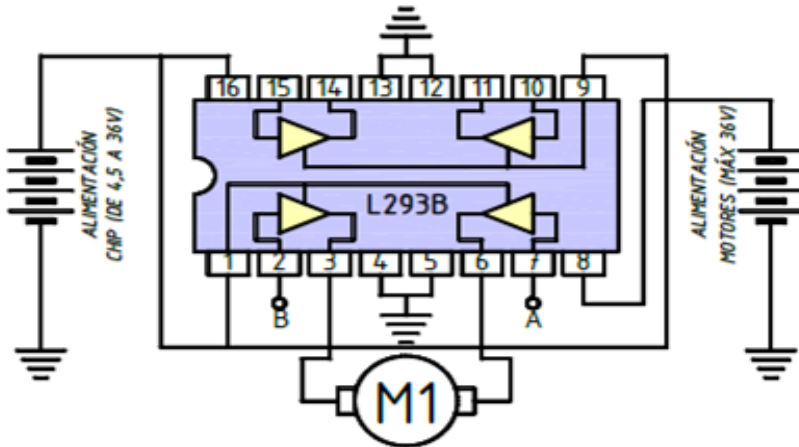


A	B	M1
0	0	PARO
1	1	PARO
0	1	IZQUIERDA
1	0	DERECHA



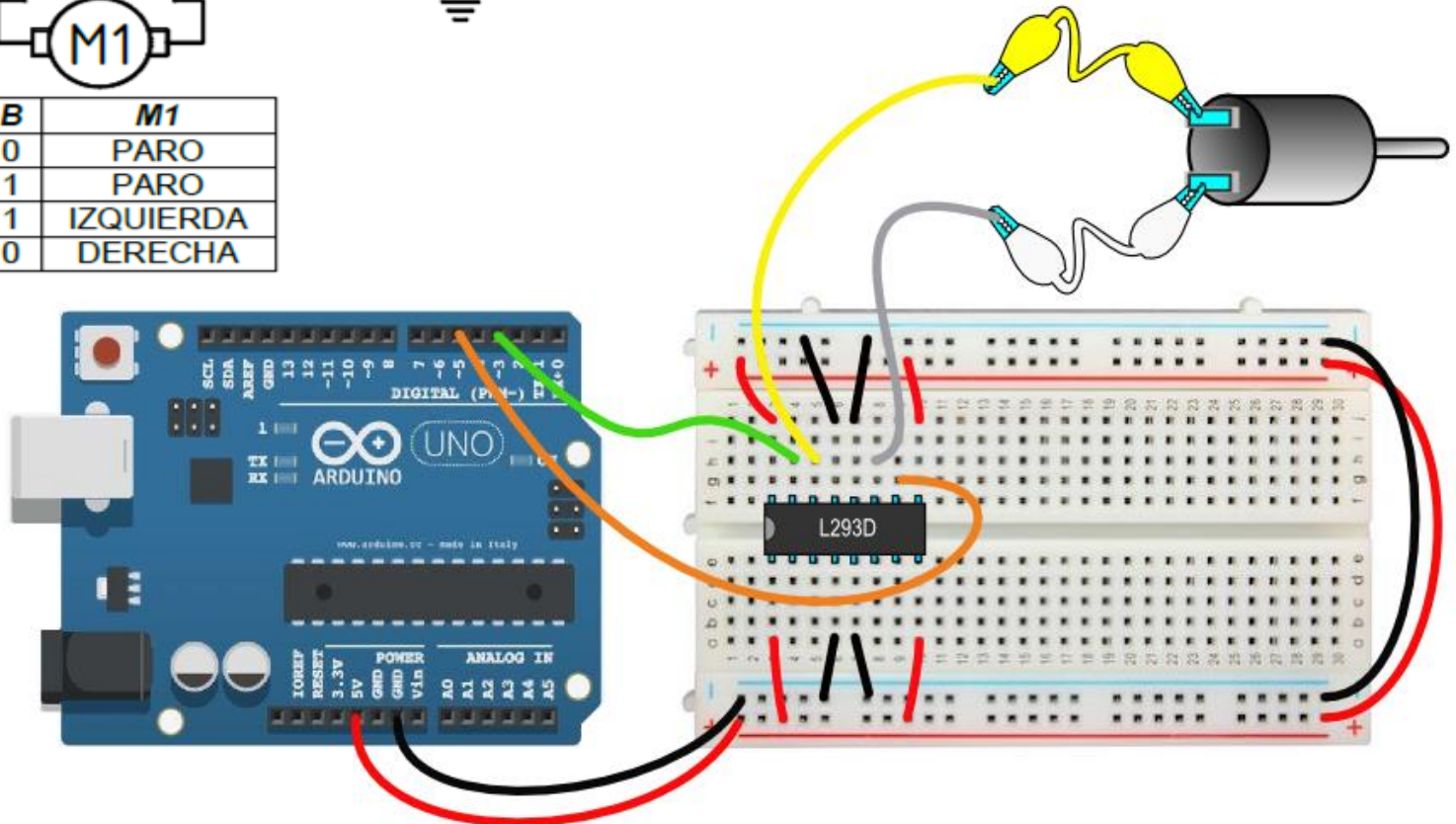
A	B	M1	C	D	M2
0	0	PARO	0	0	PARO
1	1	PARO	1	1	PARO
0	1	IZQUIERDA	0	1	IZQUIERDA
1	0	DERECHA	1	0	DERECHA

Motor de corriente continua_4



A	B	M1
0	0	PARO
1	1	PARO
0	1	IZQUIERDA
1	0	DERECHA

Además, controlando A y B con pines de salida analógica, puedo controlar la velocidad del motor.

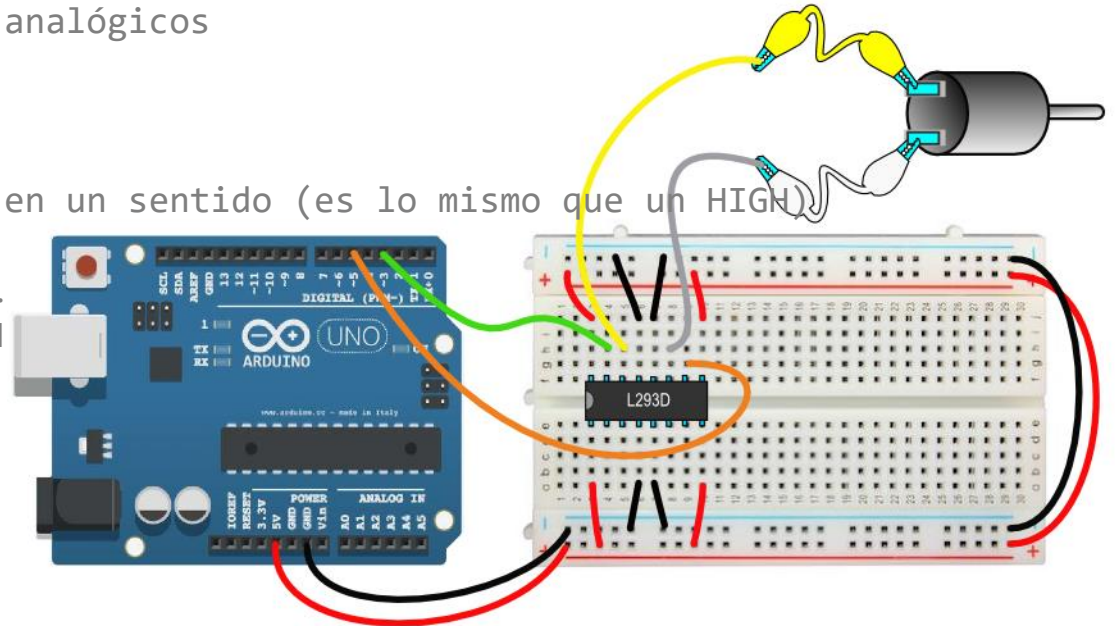


Ejemplo: Rápido, pausa y despacio



Controlaremos un motor DC con dos pines analógicos, para así controlar su velocidad.

```
void setup() {  
  pinMode(3, OUTPUT);  
  pinMode(5, OUTPUT); //pines analógicos  
}  
  
void loop() {  
  analogWrite(3, 255); //giro en un sentido (es lo mismo que un HIGH)  
  delay(1000);  
  analogWrite(5, 255); //paro..  
  delay(1000); //ambos en HIGH  
  analogWrite(3, 0);  
  //giro en el otro sentido  
  delay(1000);  
  analogWrite(5, 0); //paro..  
  delay(3000); //ambos en LOW  
  //ahora despacito  
  analogWrite(3, 150); //giro en un sentido, pero a menor velocidad  
  delay(2000);  
  analogWrite(3, 0); //paro porque están ambos en LOW  
  delay(1000);  
  analogWrite(5, 100); //giro en el otro sentido, pero a muy poca velocidad  
  delay(2000);  
  analogWrite(5, 0); //paro porque están ambos en LOW  
  delay(3000);  
}
```



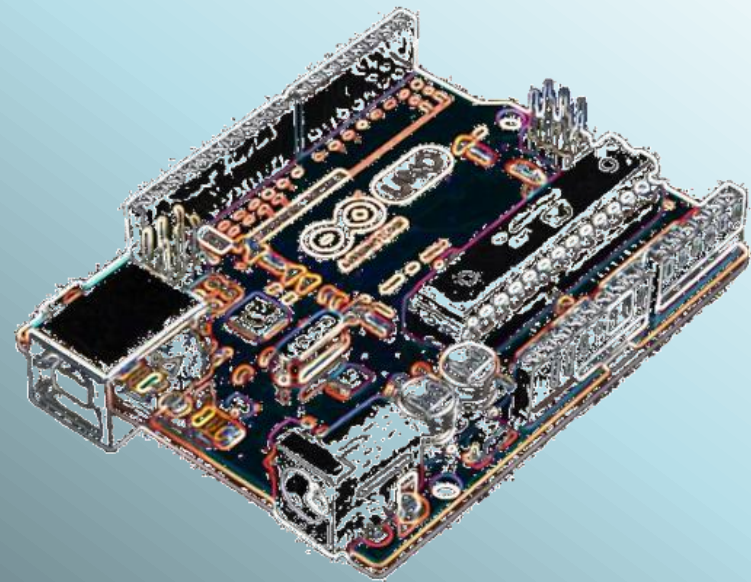
Problemas de Arduino con los motores/servomotores



Debido a que los motores son básicamente un bobinado de hilo de cobre (sin resistencia electrónica), y a su alto consumo eléctrico (sobre todo en el momento del arranque), Arduino puede presentar los siguientes problemas:

1.- Que **detecte un cortocircuito**, e inmediatamente nuestro PC **desactive** el **puerto USB** al que estaba conectado nuestra Arduino (ya no aparece nuestro COMx de la lista puertos). **Solución:** desconecta el motor/servomotor, carga el sketch a Arduino, y luego vuelve a conectarle el motor/servomotor.

2.- Que haya **excesiva demanda de corriente** por parte de los motores/servomotores y que Arduino no la pueda suministrar (sobre todo en proyectos donde movamos un vehículo con dos motores/servomotores), y eso provoque que Arduino se resetee. **Solución:** alimentar a los motores con una fuente de alimentación independiente (no a través de los pines 5 V y GND de Arduino).



Daniel Gallardo García
Profesor de Tecnología
Jerez de la Frontera
danielprofedetecno@gmail.com