Robots

Esta semana vamos a construir robots. Os juntaréis en grupos y construiréis diferentes máquinas a partir de motores. La función básica de estos motores es muy sencilla pero verás que, dependiendo de cómo los utilices, podrás animar los elementos de maneras muy diferentes. Cada grupo hará su propio robot y al final de la semana, lo enseñaréis al resto de la clase.

Antes de nada, comenzaremos con una introducción a los distintos tipos de motores que existen y veremos unos ejemplos sobre cómo utilizarlos.

# Tipos de motores

Para controlar los motores, utilizarás la placa Arduino. Básicamente, cualquier objeto electrónico que hayas visto o conoces, tiene alguna pieza en movimiento. Por ejemplo: impresoras, coches de juguete, cepillos de dientes eléctricos, etc.; todos, contienen motores. Los hay de muchos tipos, pero principalmente encontrarás tres tipos de motores:

## Motores DC (corriente continua):

Si necesitas que algo gire pero sin precisión, este es tu motor. Para encontrar un motor DC en la vida real, busca el ventilador que se encuentra dentro de tu ordenador. También puedes encontrar uno muy pequeño en tu teléfono móvil. El motor DC es el que hace que tu teléfono vibre, haciendo girar un bloque de metal cuyo peso está distribuido de manera no uniforme.



Un motor DC puede funcionar libremente en ambas direcciones, es muy fácil controlar su velocidad pero no su posición. Tampoco es sencillo hacerlo parar de forma precisa. Viene con dos cables: alimentación y tierra. Por regla general un motor DC no puede ser alimentado directamente de la corriente proporcionada por un Pin digital de Arduino.

Los cables se pueden conectar a tierra o a un Pin digital. Para hacerlo girar, establece el Pin digital en HIGH y para que se detenga, pon el Pin digital en LOW. Para hacerlo girar en sentido contrario, cambia el orden de los cables.

Es posible controlar la velocidad de un motor DC desde Arduino con una técnica llamada PWM usando un transistor. Con varios transistores dispuestos en un puente H, puedes incluso controlar la dirección sin tener que desconectar el motor.

## Motores paso a paso:

Los motores paso a paso se pueden encontrar en cualquier objeto electrónico donde prima la precisión, como escáneres e impresoras. Un motor paso a paso puede, a diferencia del motor DC, ser muy preciso tanto en posición como en velocidad.


La rotación completa de los motores paso a paso se divide en pasos equitativos y puedes controlar la parada del motor en cada uno de estos pasos. Los pasos se miden en grados, normalmente 1.8, 3.6 o 7.2. Cuanto más pequeños sean los pasos, más preciso será. Esto hace que sea muy útil cuando se necesita un posicionamiento repetido.

Sin embargo, el motor paso a paso nunca será muy rápido. Un motor paso a paso tiene 4 o más cables. Por lo general, necesitas más de 5 voltios para alimentar un motor paso a paso, lo que significa que no se puede alimentar directamente desde Arduino. Sin embargo, podemos utilizar una fuente de alimentación externa para alimentar el motor y controlarlo desde Arduino a través de un transistor.

## Servomotores

Los servomotores son ampliamente utilizados en robótica y en radio-control. Estos tipos de motores son los que vas a utilizar esta semana puesto que son muy sencillos de controlar y conectar desde Arduino.

Tienen tres cables: uno para la energía, uno para tierra y otro para controlarlos. Hay dos tipos de servomotores: rotación estándar y rotación continua. El estándar puede girar 180 grados y puede ser controlado como el motor paso a paso a una posición precisa. La rotación continua puede, al igual que el motor DC, rotar en ambas direcciones, no tan rápido; pero puedes controlar tanto la velocidad como la dirección sin tener que utilizar transistores.



﻿

# ﻿Servo estándar

Comencemos por comprobar cómo funciona el servomotor estándar.



Conéctalo al pin D9.



Para controlar los servos vamos a utilizar una librería llamada Servo. Una librería es un fragmento de código que ya ha sido escrito y que se puede instanciar (llamar) cada vez que necesites utilizarlo. En lugar de tener que escribir todo el código, solo tienes que añadir esta línea de código #include <Servo.h> a tu programa.

Escribe el siguiente en el Arduino IDE:

|  |  |
| --- | --- |
| 1234567891011121314 | #include <Servo.h> Servo myservo; void setup() {  myservo.attach(9);} void loop() {  myservo.write(0);  delay(1000);  myservo.write(180);  delay(1000);} |

Estos son los diferentes comandos que utilizamos:

* #include <Servo.h>: Incluye el código que realmente utilizamos para controlar los motores.
* Servo myservo: Crea un objeto servo myservo. Este es el que utilizas para controlar el motor.
* myservo.attach(pinNumber): Aquí es dónde indicas a qué Pin conectas el servo.
* myservo.write(degrees): Ordena al servo a qué posición girar. Como solo puede girar 180 grados, este valor varía de 0 a 180.

Lo que hace el programa es girar el servo a una posición de 0 grados, espera 1 segundo y luego lo gira a una posición de 180 grados.

## ¡Sigue experimentando!

* Cambia el ángulo del servo cambiando el valor dentro de la función write. Prueba añadir más posiciones al programa. Observa qué sucede si cambias el tiempo de retardo.
* Como puedes ver, los movimientos del servo son muy repentinos, especialmente si hay un gran salto entre dos posiciones. Esto se debe a que el servo cambia la posición prácticamente de forma instantánea. ¿Tienes alguna idea de cómo hacer los movimientos más suaves? En lugar de hacer que el servo salte directamente de una posición a otra, puedes aumentar o disminuir la posición poco a poco.
* Intenta escribir un programa que haga que el servo haga un barrido de un lado a otro, de una manera suave. (SUGERENCIA: mira el ejemplo Archivo -> Ejemplos -> Servo -> Sweep)

# Servo de giro continuo

El servo continuo tiene el mismo aspecto que un servo estándar, pero es capaz de girar continuamente como un motor normal. No puedes controlar qué ángulo está señalando pero puedes especificar la rapidez con la que quieres que gire y en qué dirección debe girar.

## https://farm6.staticflickr.com/5488/12174104774_a89e238bcc_o.png

## Materiales



* 1 placa Arduino Uno
* 1 shield Educativa
* 1 servo continuo

## Instrucciones

1. Conecta el servo al puerto D9.



|  |  |
| --- | --- |
| 1234567891011121314 | #include <Servo.h> Servo myservo; void setup() {  myservo.attach(9);} void loop() {  myservo.write(0);  delay(1000);  myservo.write(180);  delay(1000);} |

Este programa hará que el servo gire a una velocidad de 120. Puedes utilizar una velocidad entre 0 y 180. 180 es la velocidad más rápida en una dirección y 0 la más rápida también pero en la dirección opuesta. 90 debería dejarlo parado. Este valor no es exacto y puede variar de un servo a otro, por lo que deberás calibrarlo.

## ¿No funciona?

1. Asegúrate que has conectado el servo al mismo pin que el utilizado en el código.

# Servo controlado con entrada

Vamos a intentar utilizar una entrada para controlar el servo. En este ejemplo vamos a utilizar un potenciómetro.



Conecta el Pin central del potenciómetro a la entrada analógica A0. Conecta uno de los otros Pins a 5V y el último Pin a tierra. Conecta un servo estándar al pin D9.



Carga el siguiente código a tu placa Arduino:

|  |  |
| --- | --- |
| 1234567891011121314151617 | #include <Servo.h> Servo myservo;int potPin = A0;int val; void setup() {  myservo.attach(9);} void loop() {  val = analogRead(potPin);  val = map(val, 0, 1023, 0, 179);     myservo.write(val);  delay(15);} |

A este ejemplo le hemos añadido:

* Dos variables: potpin es el número del pin utilizado para el potenciómetro y val para el valor de lectura desde el potenciómetro.
* En loop() comenzamos leyendo el valor del potenciómetro.
* Los valores que leemos del potenciómetro son 0 a 1023 pero necesitamos valores de 0 a 180 para controlar el servo. Para ello, utilizamos la siguiente función:
map(value, fromLow, fromHigh, toLow, toHigh)

para ajustar los valores al rango que podemos utilizar con el servo. Esto significa que si leemos 1023 del potenciómetro obtendremos 180, si leemos 511 obtendremos 90, etc.

* Utilizamos una variable (val) para establecer la posición del servo.
* Utilizamos un pequeño delay para darle tiempo a girar al servo.

## ¡Sigue experimentando!

* Cambia el servo estándar con uno de giro continuo. Utiliza el mismo código pero suprime la función delay, cárgalo y vuelve a intentarlo. ¿Cómo actúan los diferentes servos y por qué? Piensa por qué la función de delay es necesaria cuando se controla el servo estándar pero no lo es con el continuo.
* ¿Qué más puedes utilizar para controlar los servos y qué puedes añadirles para que sean más útiles?

﻿**Utilizando dos servos**

Cuando utilices dos servos a la vez, tu placa Arduino puede tener un problema con darles la suficiente corriente. Hay diferentes técnicas para controlar esto. Aquí vas a explorar la técnica de escribir señales a los servos por separado. Este control lo harás desde tu programa.

Conecta dos servos a la shield, uno a D9 y otro a D10.




Escribe el siguiente código en el IDE de Arduino:

|  |  |
| --- | --- |
| 123456789101112131415161718192021222324 | #include <Servo.h> Servo myservo1, myservo2; void setup() {  myservo1.attach(9);  myservo2.attach(10);} void loop() {  myservo2.detach();  myservo1.attach(9);  myservo1.write(70);  delay(1000);  myservo1.write(120);  delay(1000);   myservo1.detach();  myservo2.attach(10);  myservo2.write(70);  delay(1000);  myservo2.write(120);  delay(1000);} |

El programa desconecta myservo2 cuando va a utilizar myservo1. Hace girar myservo1 un ángulo de 70 grados, espera 1 segundo y luego gira el mismo motor a 120 grados.

Del mismo modo, para hacer el mismo movimiento con myservo2 necesitas desconectar mysevo1 y conectar myservo2.

## ¡Sigue experimentando!

* Inventa un robot que use un motor de giro continuo y otro de giro estándar para que se mueva hacia adelante, ¿cómo lo harías?