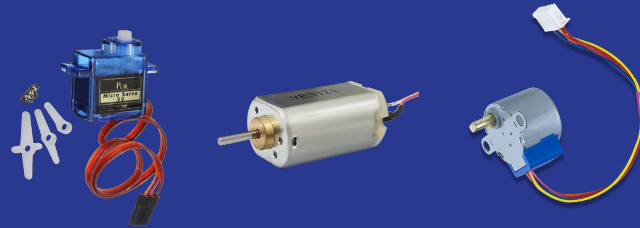


# Motores [eléctricos]

*Primeros pasos  
y perspectivas de aplicación.*

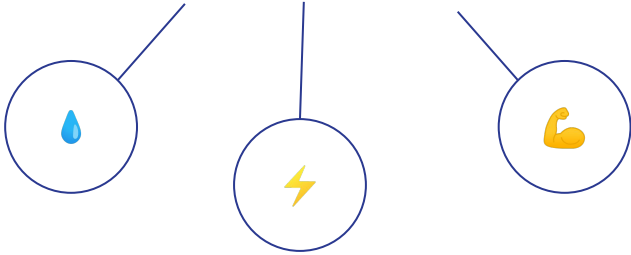


# Para empezar:

Dos [o tres] abstracciones útiles



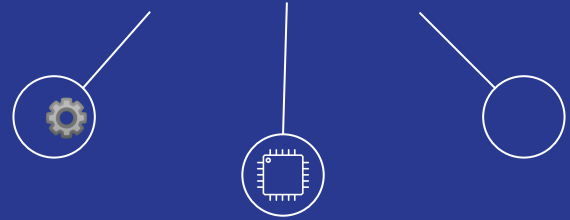
# Actuador



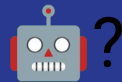
&

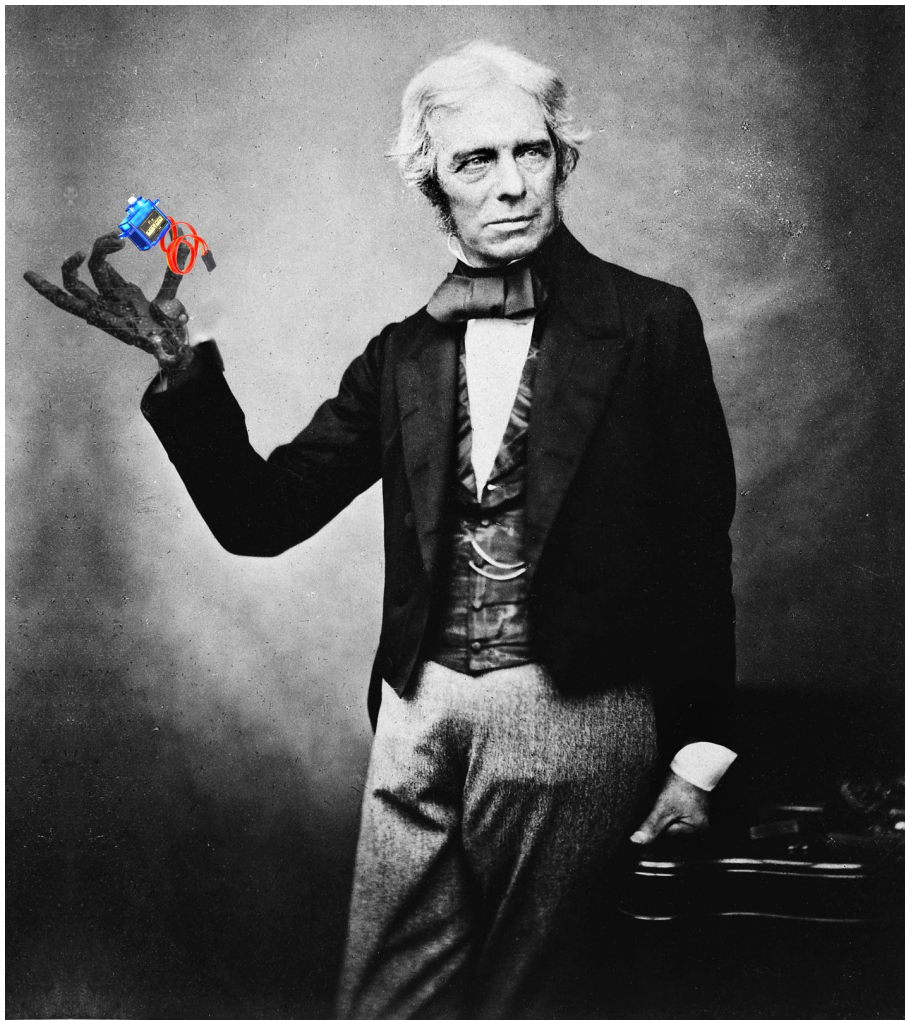


# Control



Dos [o tres] abstracciones útiles



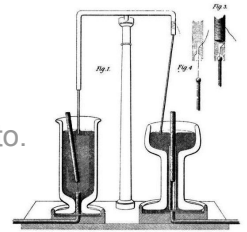


*The first demonstration of the effect with a rotary motion was given by Michael Faraday in 1821.*

Algo que giraba hace 200 años.

Muchísimas personas trabajaron en esto.

[en.wikipedia.org/wiki/Electric\\_motor](https://en.wikipedia.org/wiki/Electric_motor)



---

En el flyer:

¿Hay actuadores?

¿O controladores?

¿O sensores?

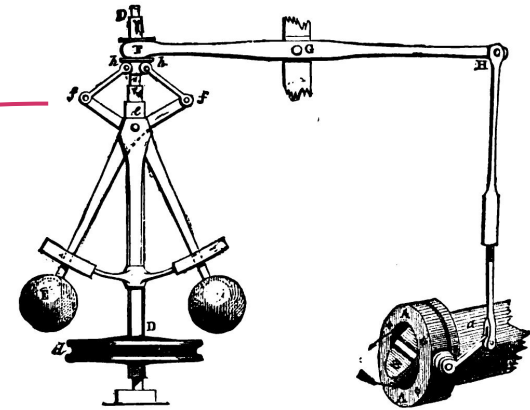
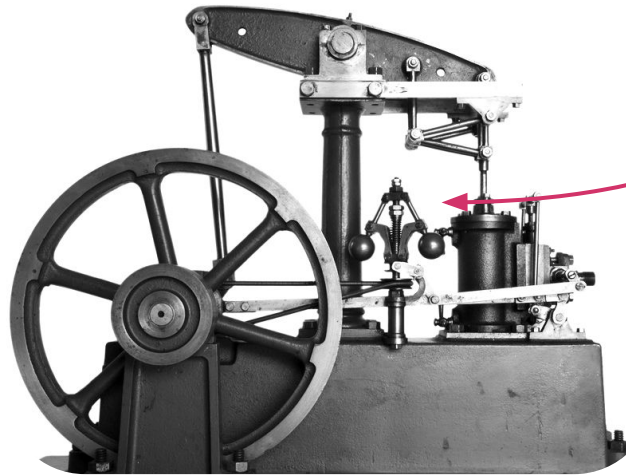
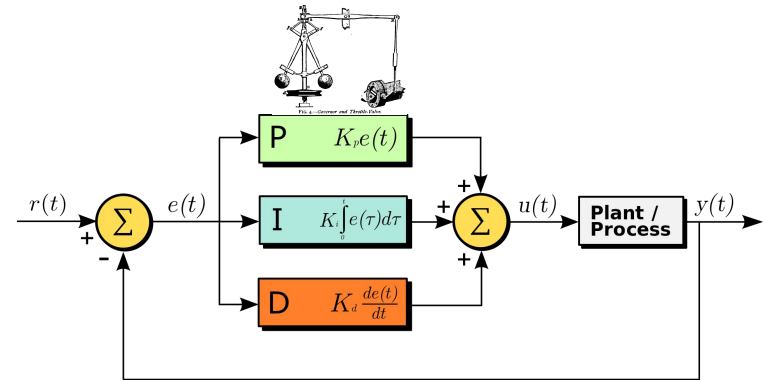
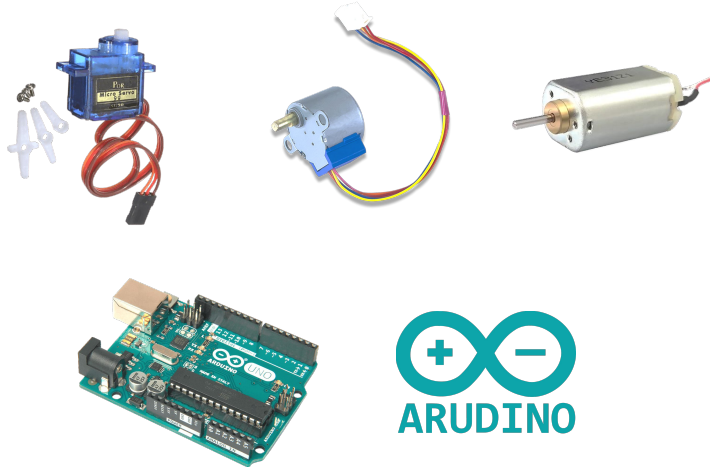


FIG. 4.—Governor and Throttle-Valve.

---

Control y actuación mecánica.  
Programabilidad *limitada*.



---

Control y actuación electrónica.  
Muy programable, muy modificable, y bastante accesible.

# Propuesta

Les propongo que vayan pensando **proyectos** en los que podrían usar un motor.

¿En qué situación querían *actuar* de forma *controlada* y/o autónoma?

# 3 tipos de motores (actuadores)

y tipos de control



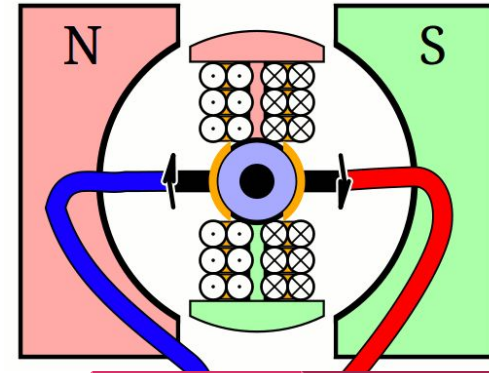
# Motores eléctricos $\Leftrightarrow$ fuerzas entre *[electro]*imanes

Vale para todos:

Motor DC, Servomotor, Motor paso a paso (PAP).

Se usan pares de campos magnéticos para generar fuerza.

Al menos uno de ellos es generado por una corriente eléctrica (es un electroimán).

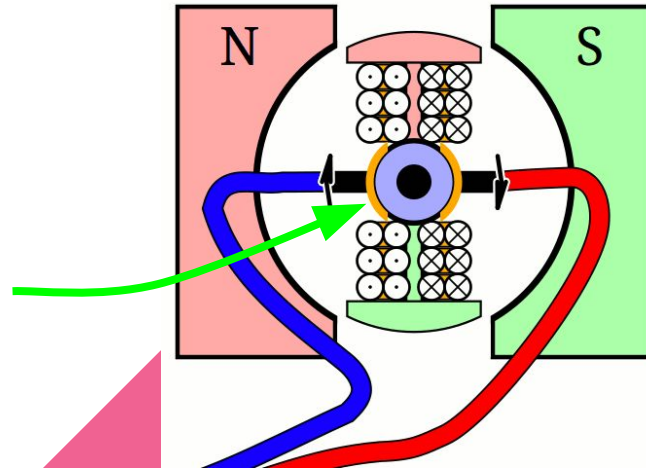


# Motores eléctricos y control

La fuerza de la actuación es electromagnética.

El control del actuador se reduce a coordinar, correctamente, corrientes a través de electroimanes.

En el caso del motor DC, la alternancia de polaridad del electroimán se logra con un conmutador **mecánico**.



# Motor DC

con ejemplos de control y aplicación



# Motores DC

“DC” porque usan corriente directa/continua.

Son más simples de mover:

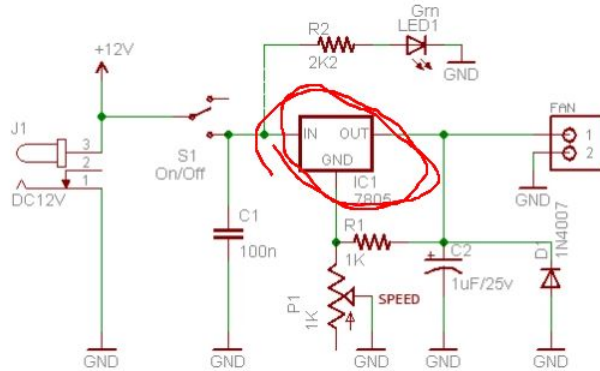
Conectar sus dos cables,  
uno a cada lado de una pila.

Y hacer girar algo :)



Son más difíciles de controlar.

The speed controller is centered around the popular 3-terminal fixed voltage regulator 7805 (IC1). The output voltage of IC1 is referenced to its common terminal which is usually grounded. In principle, the output voltage of IC1 can be raised above the designed value by simply biasing the common terminal with a suitable voltage making it easy to obtain odd-ball output voltages.



7805 voltage regulator datasheet

[All](#)
[Images](#)
[Videos](#)
[Shopping](#)
[News](#)
[More](#)
[Tools](#)

About 252,000 results (0.42 seconds)

Ad · <https://www.octopart.com/>

**LM7805CT ON Semiconductor - Voltage Regulators - Linear**

Compare Pricing, Inventory and **Datasheets** for Millions of In-Stock Parts. Octopart Is The Preferred Search Engine for Electronic Parts. Compare components. API integrations. Find electronic parts. Component parts search. Millions of instock parts. Easy to use BOM tool.

Ad · <http://www.alldatasheet.com/l7805>

**L7805 Datasheet**

L7805 **Datasheet** Search Engine. Free L7805 **Datasheet** Download. 400,000+ Searches / day. ALLDATASHEET- World's 1st. Biggest **Datasheet** Search. Update **Datasheet** Everyday. 8,000,000+ Users / Month. Distributors and Pricing. Fast & Simple Search.

Ad · <https://www.datasheetspdf.com/>

**7805 DataSheet PDF - No Registering Required**

Electronic Components **Datasheet** Search Site. Download PDF **Datasheets** for Free.

<https://www.sparkfun.com> > datasheets > LM7805 > PDF

**Voltage regulator**

This series of fixed-voltage integrated-circuit **voltage regulators** is designed for a wide range of ...  $\mu$ A7805C. TO-220, short shoulder (KCS). Tube of 20.  $\mu$ A7805CKCS ... products and disclaimers thereto appears at the end of this **data sheet**.

12 pages

# Motor DC *simple*: Pros y contras

## Pros:

- Simple.
- Barato.
- Gira rápido.

## Contras:

- Difícil controlar su posición, velocidad, o torque.
- Los “cepillos” se gastan.
- Otros motores DC son más complicados.



## Conexiones y código

```

/* DEFINIR VARIABLES */

int potPin = A0;
int motorPin = 9;

int sensorValue = 0;
int outputValue = 0;

/* CONFIGURAR ARDUINO */

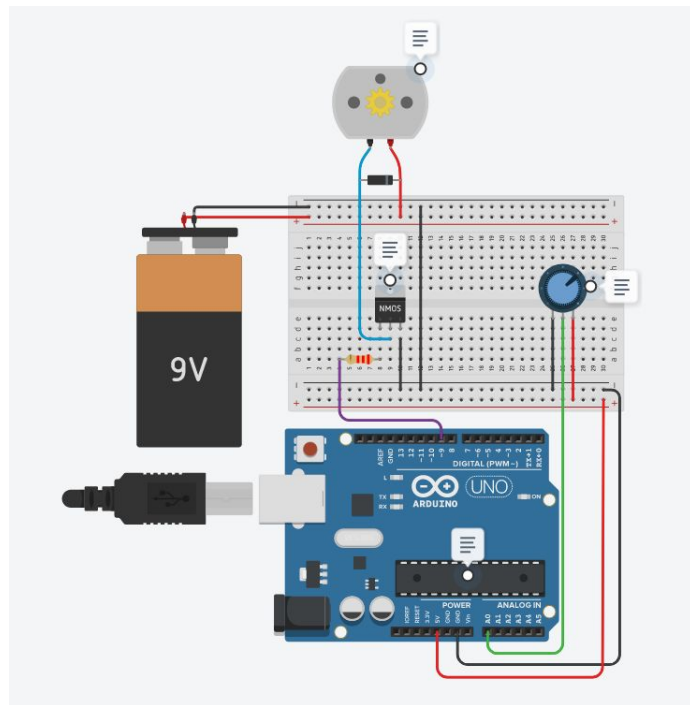
void setup()
{
    pinMode(potPin, INPUT);
    pinMode(motorPin, OUTPUT);
    Serial.begin(9600);
}

/* PROGRAMAR EL CONTROL */

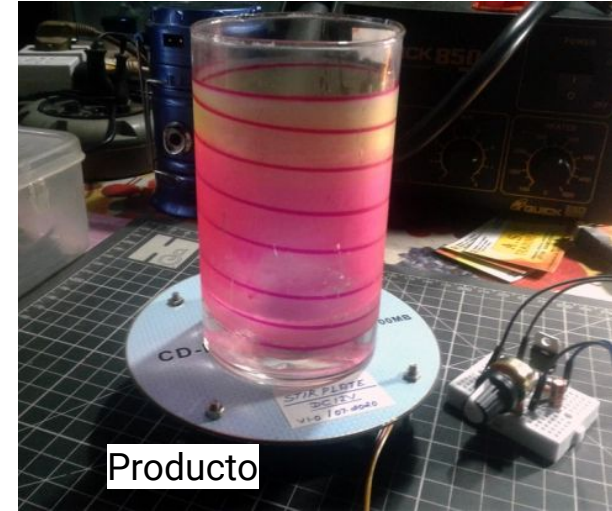
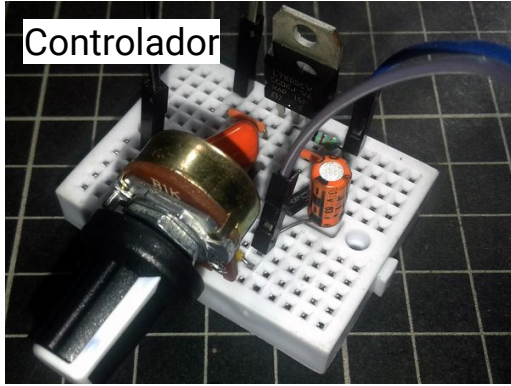
void loop()
{
    // read the value from the sensor
    sensorValue = analogRead(potPin);
    outputValue = map(sensorValue, 0, 1023, 0, 255);
    analogWrite(motorPin, outputValue);

    delay(100); // Wait for 100 millisecond(s)
}

```



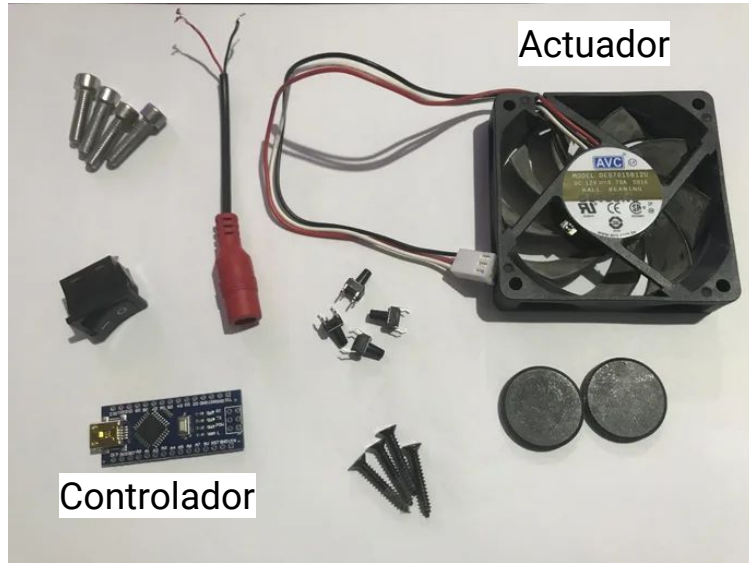
# Aplicación: control analógico (open-loop)



<https://www.electroschematics.com/stir-plate/>



# Aplicación: control digital (open-loop)



<https://www.instructables.com/Arduino-Controlled-Magnetic-Stirrer/>

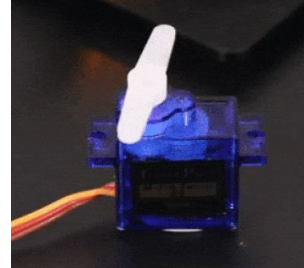
# Motor “Servo”

con ejemplos de control y aplicación



# Servomotor

En general, tienen bastante fuerza (torque)  
son más lentos, se controlan digitalmente,  
y giran entre  $0^{\circ}$ - $180^{\circ}$  grados\*.



“Servo” viene de *servomecanismo*, que supuestamente viene de “motor esclavo”.

¿Qué motor es? ¿Esclavo de qué?

# Servomotor

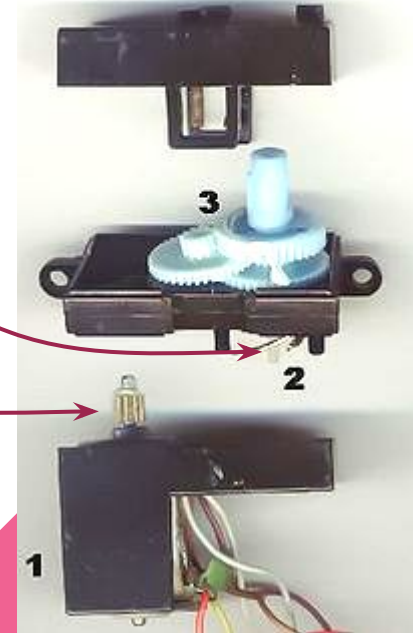


Dentro del servo hay un **motor DC** y un **circuito de control**.

El circuito lee un sensor para conocer la posición actual, 👁👁 la compara con la deseada, y controla un motor DC para achicar la diferencia.

Ese tipo de control por *feedback* se conoce como control *closed-loop*.

El motor *sabe* dónde está, y nosotros solamente le *decimos* dónde *ir*.

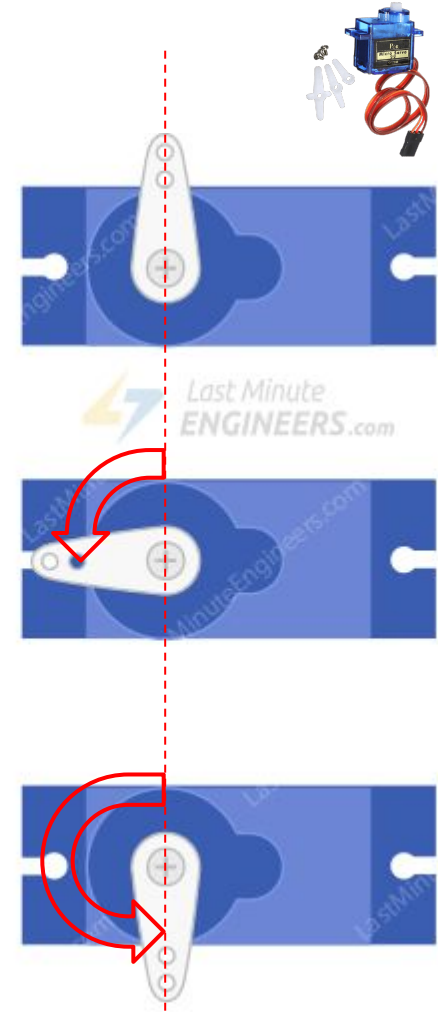
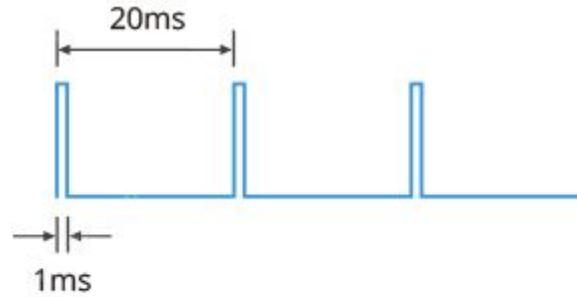


# Servomotor

Sólo necesitamos ocuparnos de **indicarle la posición deseada**.

En muchos casos, la “señal” que reciben los servos es una señal de voltaje oscilatoria llamada “PWM”.

*Pulse-width modulation*



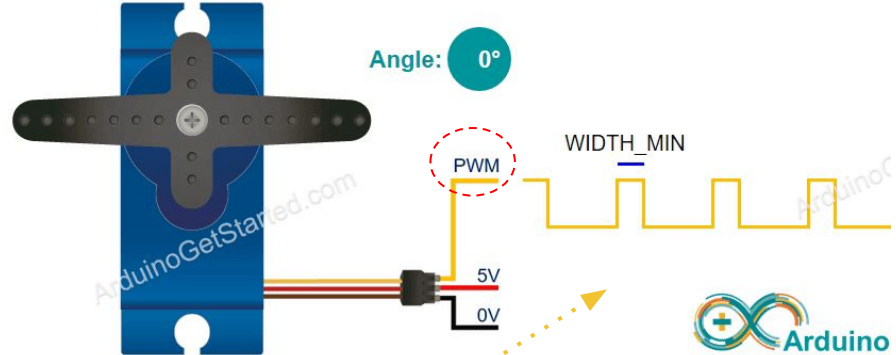
# Servomotor

Usualmente sólo necesitamos ocuparnos de **indicarle la posición deseada**.

¿Cómo?

En muchos casos, la “señal” que reciben los servos es una señal de voltaje oscilatoria llamada “PWM”.

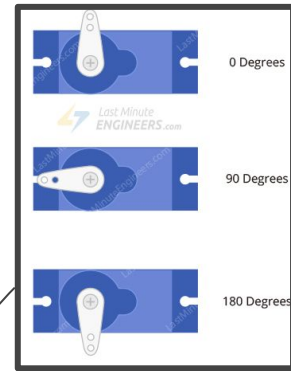
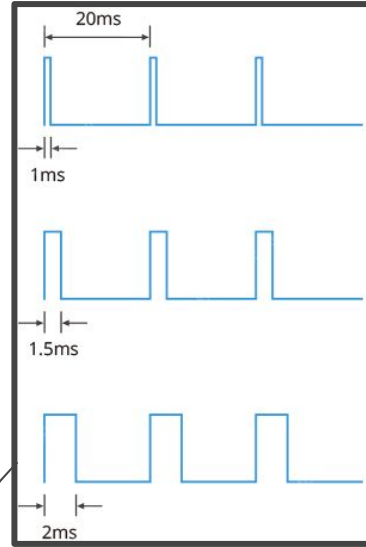
*Pulse-width modulation*



# Servomotor

La señal PWM se puede generar con un Arduino.

Y es trivial gracias a las librerías disponibles.



<https://www.arduino.cc/reference/en/libraries/servo/>

# Conexiones y Código

```
/* CARGAR UNA LIBRERÍA "Servo.h" PARA USAR EL SERVO */

#include <Servo.h>

/* DEFINIR VARIABLES DEL PROGRAMA */

int potPin = 0;    // Equivalente a "A0"
int servoPin = 9;  // Pin (tipo PWM) donde conectamos el servo.
int sensorValue;   // Para guardar la señal del potenciómetro.
int valorAngulo;   // Para guardar el ángulo.

Servo mi_servo;

/* CONFIGURAR EL ARDUINO */

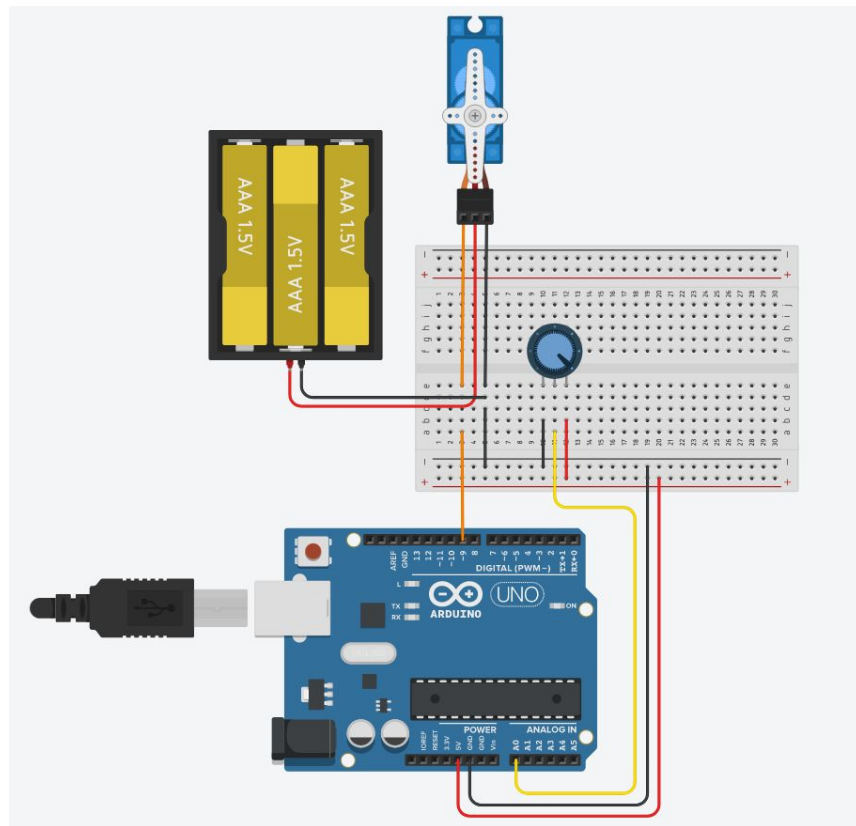
void setup () {
  pinMode(potPin, INPUT);
  mi_servo.attach(servoPin); //conecta o objeto servo1 ao pino 9
  Serial.begin(9600);
  Serial.println("Hola humano!");
}

/* PROGRAMAR EL CONTROL */

void loop(){
  sensorValue = analogRead(potPin);
  valorAngulo = map(sensorValue, 0, 1023, 0, 179);

  mi_servo.write(valorAngulo);

  delay(50);
}
```





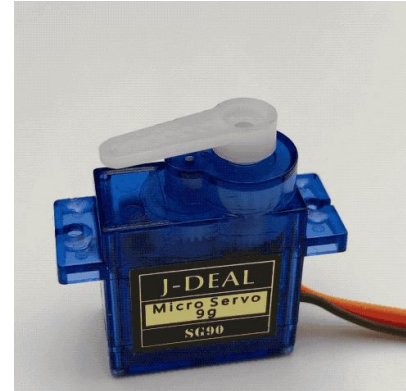
# Servomotor: Pros y contras

## Pros:

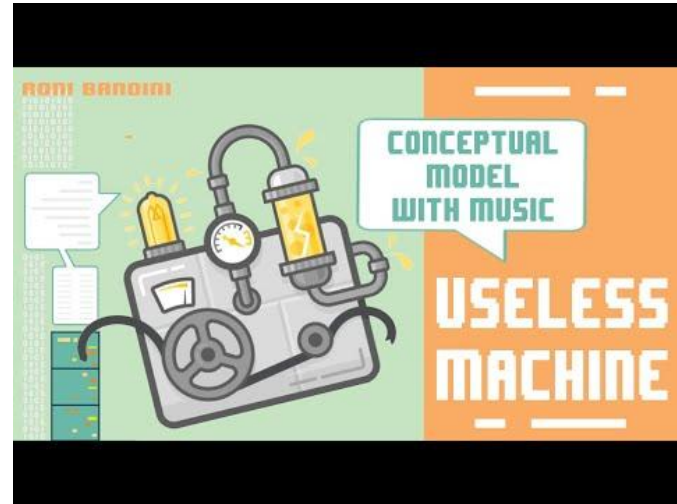
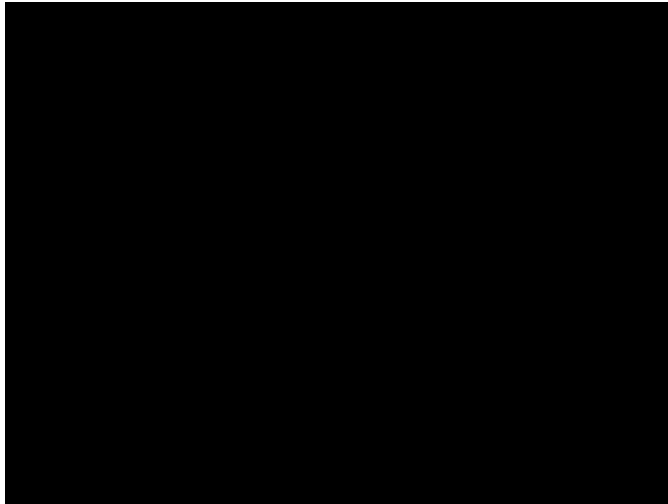
- Control de posición (closed-loop 🧐).
- Algunos tienen rotación continua (dan la vuelta y siguen).

## Contras:

- Precisión baja (en los comunes/baratos).
- Hacen más ruido.



# Aplicación: control **digital** (closed-loop)



<https://create.arduino.cc/projecthub/projects/tags/servo>

<https://create.arduino.cc/projecthub/millerman4487/simple-record-and-playback-0bffa1>

<https://www.youtube.com/watch?v=bLnAJ-mSEIE>

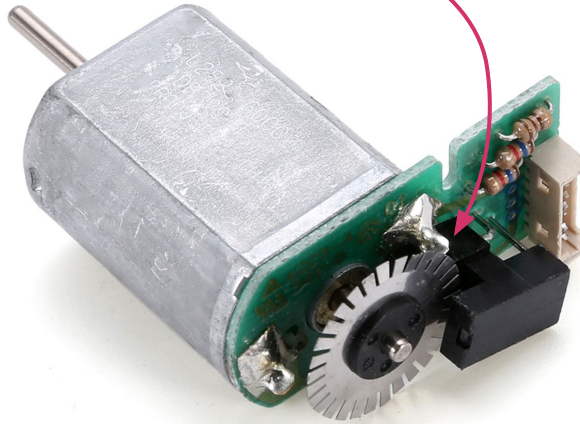
# El motor DC con encoder

Son motores DC con un cuentavueltas. 🙄

Necesitan un sistema de control externo para ser útiles.

Se podría armar un servo con uno de estos y un Arduino.

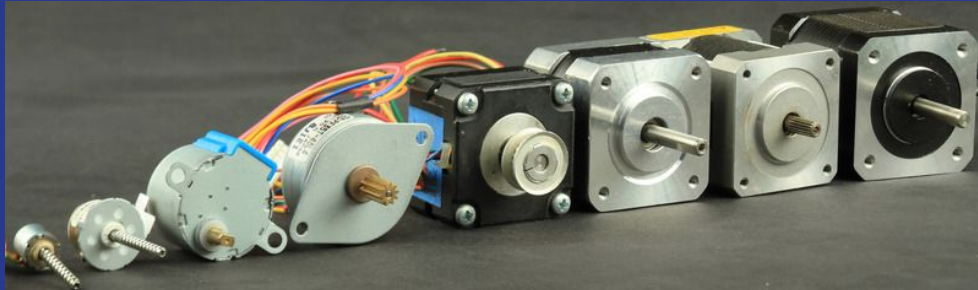
Girarán rápido, con poco torque, y posición conocida.



Disco del encoder  
(sin el sensor)

# Motor “Paso a paso”

con ejemplos de control y aplicación

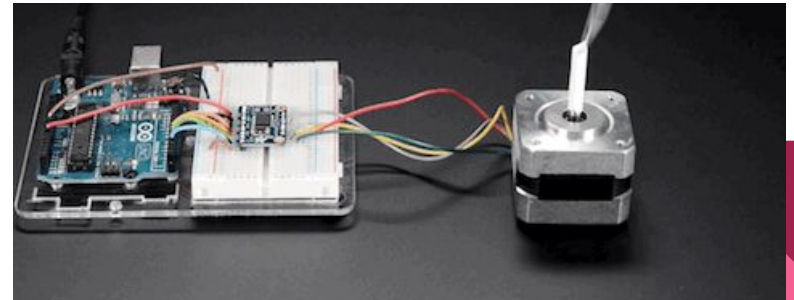


# Motor paso a paso

Vienen en variedad de tamaños, pasos por vuelta, torques, y cantidad de cablecitos para conectar.



Puede girar en ángulos exactos, en ambas direcciones, o dar muchas vueltas.

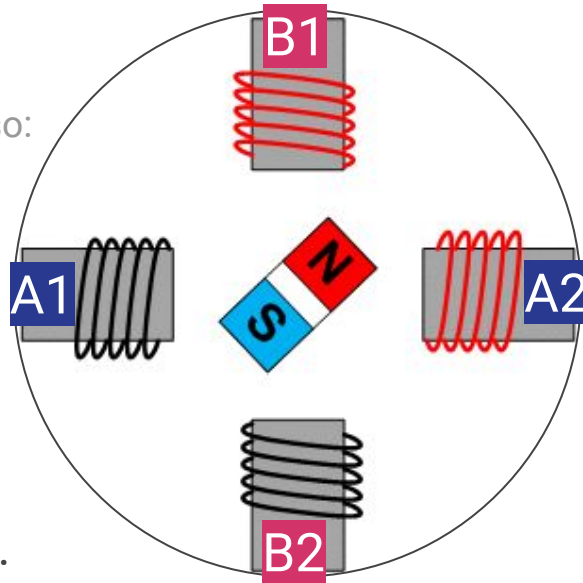


# El principio básico es casi igual

Se usan **múltiples** pares de campos magnéticos para generar fuerza:

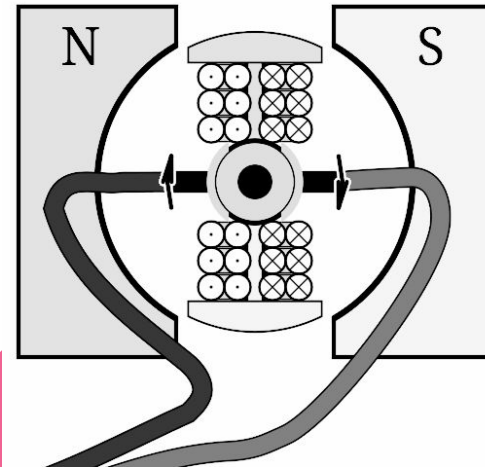
Un motor paso a paso:

- De 4 pasos
- Bipolar



La diferencia es que  
que gira “de a pasos”.

Motor DC:



# Conexiones

```
#include <Stepper.h>
const int pasosPorRev = 4;
const int velocidadRPM = 16;

int coilA1 = 9;
int coilA2 = 10;

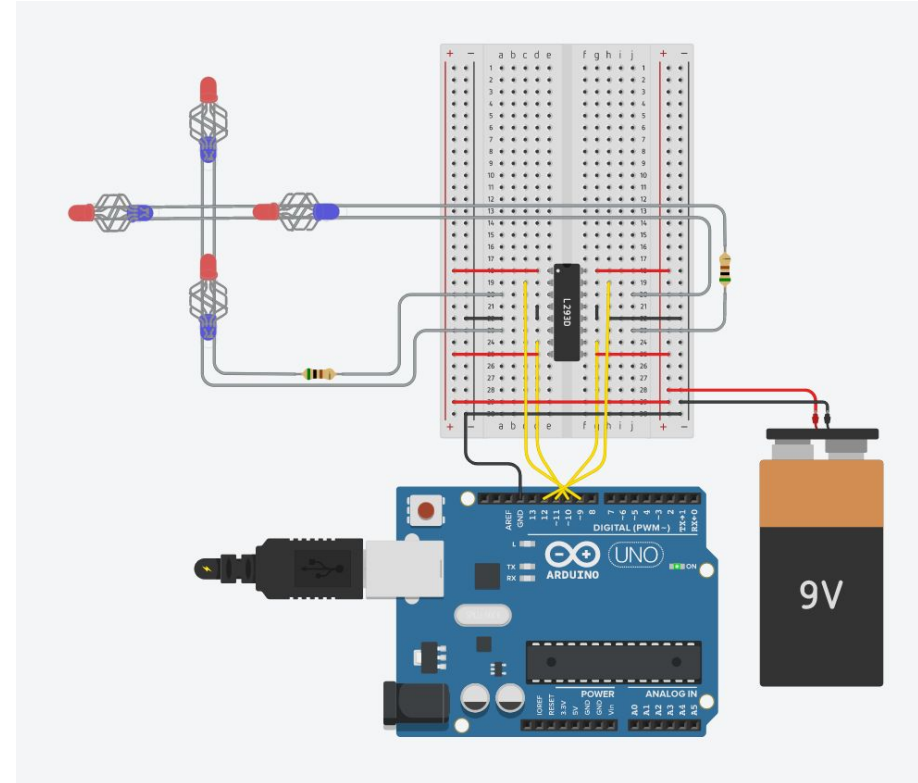
int coilB1 = 11;
int coilB2 = 12;

Stepper myStepper(pasosPorRev, coilA1, coilA2, coilB1, coilB2);

void setup()
{
  myStepper.setSpeed(velocidadRPM);
  Serial.begin(9600);
}

void loop()
{
  myStepper.step(pasosPorRev);
  Serial.println("Sentido horario");
  delay(1000*30);

  myStepper.step(-pasosPorRev);
  Serial.println("Sentido anti-horario");
  delay(1000*30);
}
```



# Código





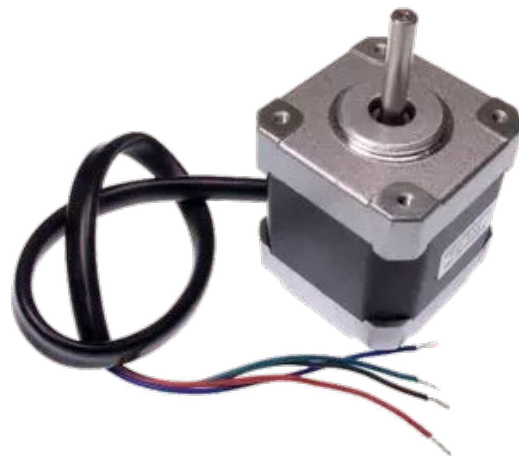
# Motor paso a paso: Pros y contras

## Pros:

- Control de posición (open-loop 🦊).
- Muy precisos (+microstepping).

## Contras:

- Electrónica dedicada para mover y controlar.
- Pueden perder pasos si la carga es alta,
- y/o si la velocidad de rotación es demasiada.



# Pérdida de pasos

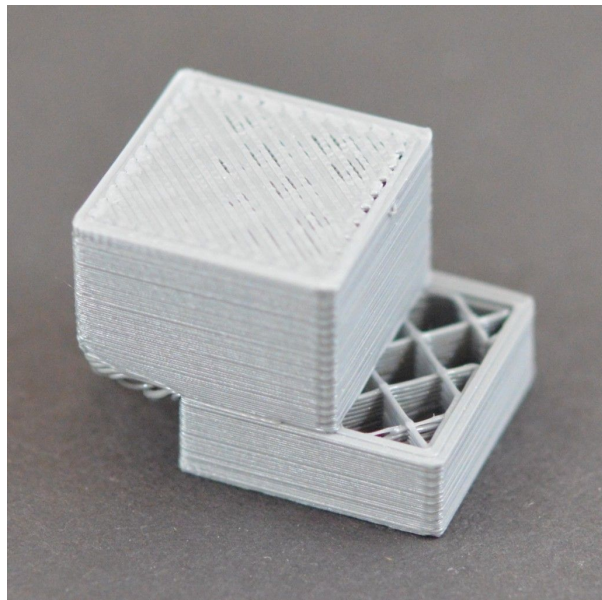
Si un motor paso a paso encuentra demasiada resistencia o se le pide demasiada velocidad, puede perder pasos.

Pero, como no lo podemos medir, esto hace que inadvertidamente se pierda el sistema de referencia:

*Pensamos que llegó donde queríamos,  
pero no es así: se quedó corto.*

Si la tarea depende de que el motor se haya movido exactamente lo que el programa indicó...

Entonces es **muy** importante no perder pasos.

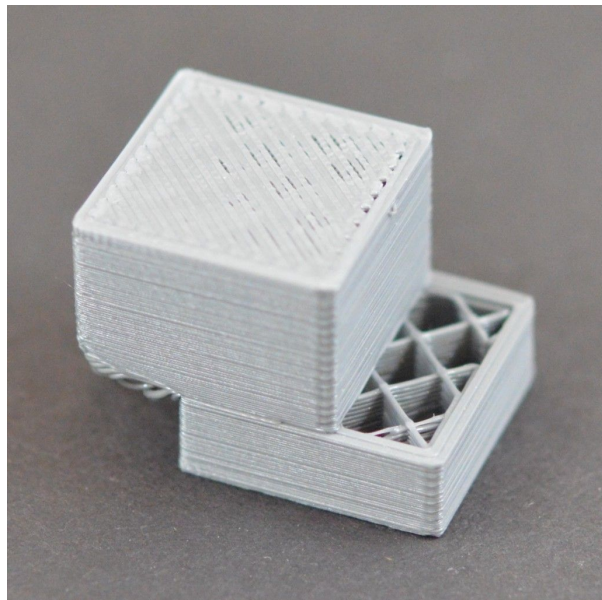


*Layer-shift* en la impresión 3D de un cubo.

# Pérdida de pasos

La solución más sencilla es:

- elegir el motor paso a paso correcto,
- suministrarle la corriente que requiere,
- y respetar la carga y la velocidad máxima.



*Layer-shift* en la impresión 3D de un cubo.

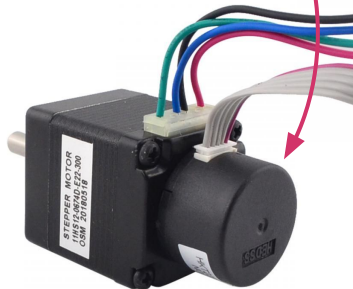
# Motor paso a paso con encoder

Son motores PAP con un cuentapasos. 🙄

Necesitan *otro* sistema de control externo para ser útiles.

Se podría armar un servo con uno de estos y un Arduino.

Son útiles cuando es muy importante saber si se pierden pasos (y preservar la información de posición).



Puede costar lo mismo que el motor solo.

# Práctica

**Virtual**, usando Tinkercad.

# Hardware [simulado] y Software

A diferencia de los talleres sobre software, este es sobre **hardware**.

Vamos a cubrir la parte de **control**, usando simuladores de código y componentes virtuales (en TinkerCAD).

La idea de la robótica es **actuar** sobre el mundo físico, así que les invitamos a salir del simulador y embarrarse apenas tengan la oportunidad de hacerlo ;)



# Como usar TinkerCAD

Mini tuto



# De qué va la práctica

3 motores





# Práctica A: Consignas

1. Empezar con el primer motor.
2. Hacer conexiones.
3. Ajustar el código.
4. Simular el control con un Arduino virtual.
5. Pasar al siguiente motor :)

Para cada motor hay un espacio de trabajo de TinkerCAD Circuits y guías (en playlists de YT):

1. DC:
  - [www.youtube.com/playlist?list=PLSqqZBTIQ\\_dx6SCoEyrkmZ2SE9fj-zjI4](https://www.youtube.com/playlist?list=PLSqqZBTIQ_dx6SCoEyrkmZ2SE9fj-zjI4)
  - [www.tinkercad.com/things/3qqZ7kmtmZn-copy-of-ejercicio-01-dc-motor-mosfet-npn-v2](https://www.tinkercad.com/things/3qqZ7kmtmZn-copy-of-ejercicio-01-dc-motor-mosfet-npn-v2)
2. Servo:
  - [www.youtube.com/playlist?list=PLSqqZBTIQ\\_dx4w11xbpqlYgTck6r85yd5](https://www.youtube.com/playlist?list=PLSqqZBTIQ_dx4w11xbpqlYgTck6r85yd5)
  - [www.tinkercad.com/things/8NmQMqQeZPII-copy-of-ejercicio-02-micro-servo](https://www.tinkercad.com/things/8NmQMqQeZPII-copy-of-ejercicio-02-micro-servo)
3. PAP:
  - [www.youtube.com/playlist?list=PLSqqZBTIQ\\_dyivXVG-AC7qewWcbPNDHUV](https://www.youtube.com/playlist?list=PLSqqZBTIQ_dyivXVG-AC7qewWcbPNDHUV)
  - [www.tinkercad.com/things/24GaUqRBbyY-copy-of-ejercicio-03-control-de-motores-paso-a-paso](https://www.tinkercad.com/things/24GaUqRBbyY-copy-of-ejercicio-03-control-de-motores-paso-a-paso)


# Práctica A: Soluciones

Soluciones:

- DC:  
[www.tinkercad.com/things/au1eUH9w0h-copy-of-ejercicio-01-dc-motor-mosfet-npn](http://www.tinkercad.com/things/au1eUH9w0h-copy-of-ejercicio-01-dc-motor-mosfet-npn)
- Servo:  
[www.tinkercad.com/things/hTam56YTeRw-copy-of-micro-servo-2/](http://www.tinkercad.com/things/hTam56YTeRw-copy-of-micro-servo-2/)
- Stepper:  
[www.tinkercad.com/things/94RQacvXZHC-copy-of-stepper-motor-control/](http://www.tinkercad.com/things/94RQacvXZHC-copy-of-stepper-motor-control/)

Al final probaremos sus códigos en vivo,  
usando los motores que tengo en casa :)

Luego hablaremos un poco más relajadamente,  
sobre motores y aplicaciones, si queda tiempo.



# Final!

Probar en casa.



## Práctica B: Alternativa

Si ya tienen motores en sus casas, pueden hacer más cosas:

- Si tienen a mano todos los componentes, aprovechen para armar algo :)
- Si tienen motores rescatados tirados por ahí, podemos identificarlos y recomendar alguna forma de controlarlos.
- Si tienen ideas de proyectos, pueden consultar cómo realizarlos y pensamos qué se puede hacer.





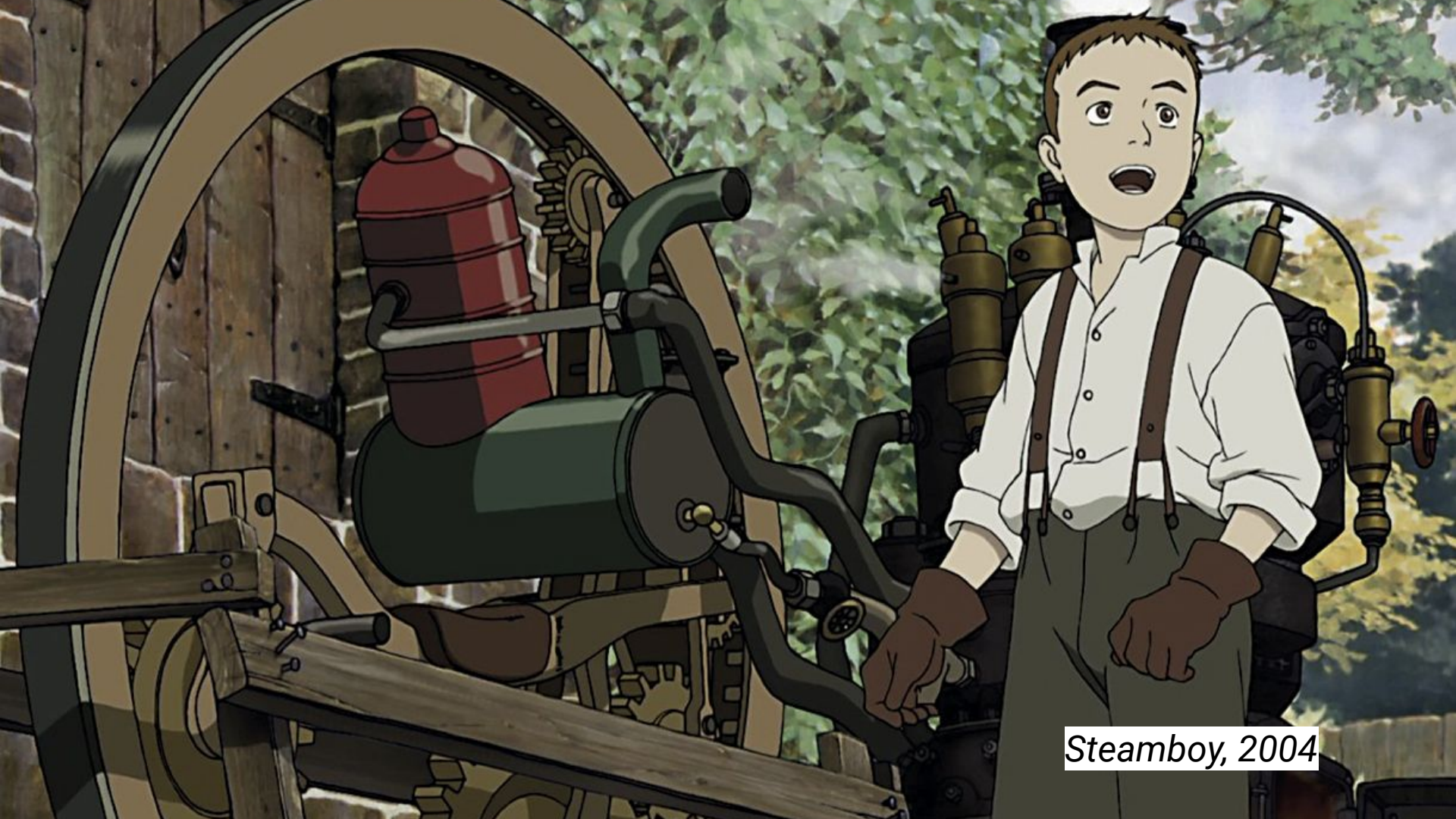
# Fin!

Gracias por su atención; espero que les resulte útil haber participado!

Pueden escribirnos con consultas y/o para ayudarles a realizar su proyecto.



BONUS: algunas pelis con motores



Steamboy, 2004





*Wild Wild West, 1999*