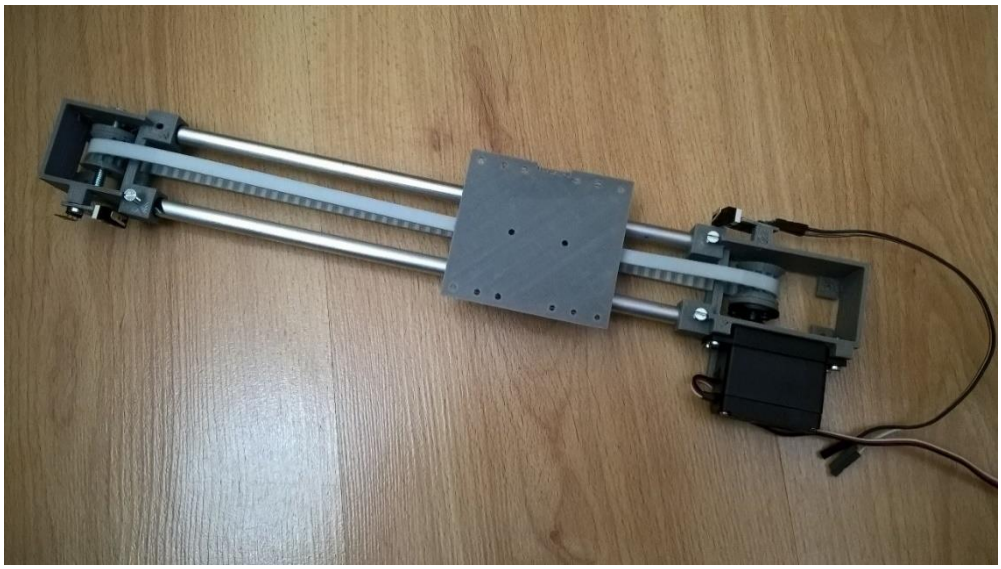
# Actuador lineal



Práctica 4 – Actividades (v1.5.1 septiembre 2022)

## Software para robots

Cristian González García

gonzalezcristian@uniovi.es

Basado en el material original de Jordán Pascual Espada

# Índice

El total de las actividades principales tienen un valor de 0,6 puntos y 0,6 puntos de ampliaciones dentro del **bloque 3**.

## Consideraciones importantes

- **Apretar los tornillos que presionan la cinta lo mínimo posible**, lo justo para que la cinta no se mueva. Si seguimos aplicando presión romperemos la pieza, así que, **¡cuidado!**
- **Apretar los tornillos que presionan la barra lo mínimo posible**, lo justo para que la barra no se mueva, pues si seguimos aplicando presión romperemos la pieza, así que, **¡cuidado!**
- **La correa debe de estar tensa**. Si no está correctamente tensada lo notaremos al utilizar el actuador, pues se moverá a tirones. Si se da este caso, hay que tensarla, pero sin pasarse, así que, **¡cuidado!**
- **Nota muy importante**: si se os queda atascado el carro en un lateral durante su funcionamiento por una mala o incorrecta programación, o veis que esto va a ocurrir, o al subir el código queda el motor haciendo fuerza y golpeando y haciendo saltar el carro y los engranajes, desenchufad rápidamente uno de los cables del servomotor para que así podáis subir el código sin que se estropee el actuador lineal, el motor, o alguno de los otros componentes. Una vez esté el código subido, enchufáis ese cable y listo. Todo tipo *plug and play*.

Los ejercicios principales dependen del primero, así que, deberéis tenerlos implementados en orden si queréis ir haciendo el siguiente. Las siguientes prácticas dependerán de varios ejercicios de esta práctica, tal y como se indica en cada ejercicio.

Lista de reproducción en YouTube:

https://www.youtube.com/watch?v=07iYgehBpTo&list=PLpe5dTI2xCy8CNbYdZPkCwvCwpLV4lbkW

## Ensamblar el actuador lineal

Tal vez se requiera que tengáis que ensamblar los componentes del actuador lineal utilizando como base el actuador lineal funcional. Tendréis que aseguraros que todos los componentes funcionan y que todo se ha ensamblado correctamente antes de comenzar.

## Actividades principales

### (Teleoperado4.1) Actuador lineal teleoperado (0,15 Puntos)

*Material necesario: 1 actuador lineal y 1 joystick.*

Se requerirá tenerlo implementado para los ejercicios principales 4.2, 4.3, 4.4, y la ampliación 4.5

Utilizar el joystick para manejar el actuador lineal.

El joystick debe controlar el movimiento del actuador. Al mantener pulsado el joystick hacia adelante, el actuador debe moverse hacia delante. Al mantener pulsado el joystick hacia atrás, el actuador debe moverse hacia atrás. Cuando el usuario suelte el joystick, el movimiento del motor debe detenerse.

**Recomendaciones:**

Mirar los valores que da el joystick en la posición por defecto, pues a lo mejor no devuelven exactamente (512, 512). En base a lo que devuelve hay que crear la lógica, siempre dejando cierto margen. Es decir, si devuelve (496, 501) hacer que se mueva hacia adelante cuando se reciba más de 505, hacia atrás cuando reciba menos 490, y esté parado en el resto de los casos. De esta manera, se tendrá calibrado el joystick por software y nos ahorraremos muchos problemas. Si se deja suficiente margen, puede servir para otros joystick que tengan más variación.

Vídeo del ejercicio: https://youtu.be/Uy1Hkpfr3nc

## (Velocidad4.2) Dos niveles de velocidad (0,15 Puntos)

*Material necesario: 1 actuador lineal y 1 joystick.*

Requiere tener implementado el ejercicio principal 4.1

Dependiendo de la orientación y profundidad que el usuario ejerza sobre el joystick, el actuador lineal se debe mover en esa orientación más o menos rápido.

Se tendrán en cuenta únicamente dos velocidades, «despacio» y «rápido», tanto hacia la derecha como hacia la izquierda. Por ejemplo, (< 100) rápido izquierda, (100 – 412) despacio izquierda, (412 – 612) parado, (612 – 913) despacio derecha, y (> 913) rápido derecha.

Es posible que estos valores no se correspondan de forma exacta con los valores del joystick que se esté utilizando.

Los valores de velocidad del servo pueden no ser lineales, es decir, 45 no es la velocidad intermedia entre 0 y 90.

Vídeo del ejercicio: https://youtu.be/SGLILywP1mE

## (Seguro4.3) Recorrido seguro (0,15 Puntos)

*Material necesario: 1 actuador lineal, 1 joystick y 2 sensores de colisión.*

Requiere tener implementado el ejercicio principal 4.1

Se requerirá tenerlo implementado para la ampliación 4.4

Hay que colocar dos sensores de colisión en las dos partes del actuador lineal y hacer que cuando choque contra ellos, deje de avanzar a pesar de que se siga enviando la orden de avanzar en esa dirección con el Joystick.

Los sensores de colisión deben detectar que el carro ha tropezado con ellos. Es necesario que se graben como ambos sensores funcionan en el vídeo.

## (Automático4.4) Recorrido automático (0,15 Puntos)

*Material necesario: 1 actuador lineal, 1 joystick y 2 sensores de colisión.*

Requiere tener implementado el ejercicio principal 4.1 y el 4.3

Se requerirá tenerlo implementado para la ampliación 4.4

Hay que utilizar los dos sensores de colisión de ambas partes del actuador lineal para hacer que el carro se mueva automáticamente de un lado a otro. La condición para que cambie de sentido se cumple cuando el carro colisiona con un sensor.

Los sensores de colisión deben detectar que el carro ha tropezado con ellos tanto en modo automático como en el modo manual. Es necesario que se graben ambos usos en el vídeo, así como que funcionan ambos sensores.

Para activar/desactivar el modo de recorrido automático se debe presionar el botón del joystick.

- Presionar una vez: activa el modo automático.
- Presionar otra vez: desactiva el modo automático y vuelve a manejarse con el joystick.

Al grabar el vídeo, se tiene que apreciar claramente que no tocan el joystick.

**Posible enfoque:**

```
int mode = 0;

void setup() {
  // Aquí el código que quieras que se ejecute 1 sola vez
}

void loop() {
  // Código que se ejecutará en cada interacción del loop
(varias veces por segundo y sin parar)
  if (mode == 1){ // Modo automático
    // Lógica del modo automático
  } else if (mode == 2){ // Modo manual
    // Lógica del modo manual
  }
}
```

Vídeo del ejercicio: https://youtu.be/KJH6FgInm9g

# Ampliaciones

## (Coordenadas4.5) Sistema de coordenadas (0,25 Puntos)

*Material necesario: 1 actuador lineal y 2 sensores de colisión.*

Requiere tener implementado el ejercicio principal 4.4

Se requerirá tenerlo implementado para las ampliaciones 4.6, 4.7, 6.1 y 6.2

Hay que establecer un sistema de coordenadas para el espacio del actuador lineal.

Normalmente, los espacios de coordenadas están basados en alguna medida estándar, como centímetros, pero en este caso utilizaremos una medida no estándar.

En primer lugar, tenemos que calcular la «**capacidad de recorrido**» que tiene el actuador en el eje X. El recorrido completo será la trayectoria desde una esquina hasta la otra. Esto lo detectamos cuando llega a una esquina porque presiona un sensor de colisión.

El programa debe iniciarse en modo «calibración», de forma que el actuador vaya desde una esquina a otra, haciendo el recorrido completo. Si no está en una esquina al inicio, deberá ir hacia una y comenzar desde ella. Para hacer esto, basta con enviar el carro hasta que choque con un sensor de colisión.

Una vez el carro esté en una esquina, debe recorrer todo el recorrido hasta que choque contra el otro sensor de colisión. Esto, internamente, debe calcular el número de milisegundos que tarda en realizar el recorrido completo, como por ejemplo pueden ser 4.500 milisegundos (ms). Este tiempo servirá para calcular cuántos ms debemos mover el servomotor para ir a una determinada coordenada.

Vamos a establecer que el recorrido tiene un total de 24 posiciones en el eje. Por ejemplo, si estamos en la coordenada 0 y hay que ir a la 3, el servomotor debería moverse (4500/24 * 3) ms.

Una vez el programa finalice la calibración, debemos hacer que el actuador se mantenga a la «espera de coordenadas».

En este primer ejercicio, **las coordenadas estarán *hardcodeadas*** en el código fuente y se le enviarán al actuador una vez termine la fase de calibración. Una coordenada negativa indicará que vaya a la 0, y una mayor de 24 indicará que irá a la 24. En ambos casos, en vez de moverse X ms, como son **las coordenadas finales e iniciales**, una posibilidad es **mover el carro hasta que choque con el sensor de colisión** correspondiente a la coordenada solicitada.

Las coordenadas se envían una detrás de otra y el actuador debe poder ir de la 0 a la 5, después a la 20, después a la 15, etc. y **moverse sin pasar por el 0**, es decir, de una a otra de forma continua y **sin reiniciar** el actuador a su posición inicial. Es decir, no vale que vaya a la 15, después al 0, después a la 7, después al 0… Esto implica que **el actuador deberá tener en memoria la coordenada en la que se encuentra** y realizar el cálculo para ir a la coordenada solicitada.

**Vídeo**: para enseñar que funciona, grabad el actuador y que se vean las coordenadas que tenéis *hardcodeadas*. Además, hacer que vaya a coordenadas sin pasar por el 0, como en el ejemplo de arriba y probad también alguna coordenada negativa y mayor del límite positivo. El carro deberá de parar en el sensor de colisión si se introduce una coordenada mayor o menor de los límites. Enseñad varias coordenadas intermedias que impliquen ir en ambos sentidos, una coordenada negativa y una mayor del límite existente.

**Notas**:

- Para iniciar la calibración se envía el carro hacia una esquina debido a que no es posible saber en qué posición del recorrido está el carro cuando encendemos el actuador. No estamos guardando esa información si se apaga el Arduino, luego, necesitamos tener una referencia. Si hubiera que guardarla, una opción sería tener una tarjeta SD se podría guardar en un fichero, o bien en un servicio en la nube. Al no disponer de ellas y saber dónde está el carro, debemos moverlo hacia una de las esquinas y detectar cuando han colisionado con ella. Es en este momento cuando podemos tomar como referencia esa posición, que será el 0.
- Cuidado con la variable que utilicéis para el tiempo. Lo mejor es que sea del tipo «**unsigned long**» (4 bytes/32 bits: -2,147,483,648 to 2,147,483,647) o «**unsigned double**» (8 bytes/64 bits), ya que solo vais a trabajar con números positivos (al ser *unsigned* tendríais el doble de números positivos) y de esta manera tendréis mucho margen hasta el *overflow*.
- **Verificar que la correa está bien ajustada** y no pega saltos ni se salta dientes, pues si está algo floja, puede saltar algún diente de vez en cuando y entonces el sistema funcionará incorrectamente. Además, irá a tirones.
- **Verificar en varias pasadas que el carro tarde aproximadamente el mismo tiempo en hacer el mismo recorrido entero**, pues puede ser que la correa no esté bien ajustada, el motor esté ya muy usado, se estén recogiendo mal los tiempos, o haya cualquier otro problema.
- **Revisar que se mueve en ambas direcciones a la misma velocidad**. Si esto no ocurriera, podría ser que el motor estuviera mal. Una solución software sería tomar el tiempo que tarda en moverse en una dirección y en la contraria, y dependiendo en qué dirección vaya, el motor tendrá que estar más tiempo funcionando o menos.
- **Llegan 0 no introducidos**: si llegan 0 después de enviar peticiones es debido a que, en el IDE del Arduino, la consola está seleccionado la opción salto de línea. Si se cambia a otra no introducirá más 0 una vez se deje de introducir datos. **¡¡¡MUY IMPORTANTE!!! Revisad el salto de nueva línea de la consola.**

## (Consola4.6) Coordenadas desde consola (0,25 Puntos)

*Material necesario: 1 actuador lineal y 2 sensores de colisión.*

Requiere tener implementado el ejercicio principal 4.4 y la ampliación 4.5

Hay que implementar un sistema para introducir coordenadas usando el teclado del PC **utilizando la consola del IDE de Arduino**. El actuador se debe colocar en la coordenada indicada. El formato de envío de coordenadas será el siguiente: «coordenada,milisegundos de espera;». Se deben poder recibir conjuntos de coordenadas separadas por puntos y coma. Ejemplo:

10,1000;20,1000;10,400;0,500;

20,1000;21,500;1,1000;-2,3000;

**Excepciones**:

- Si la **coordenada** es **mayor** que las existentes **o** es **negativa**, el carro debe ir hasta el límite del actuador y pararse.
- Si se introduce una **coordenada negativa** debería de funcionar como en el ejercicio anterior, es decir, ir al 0, y si es mayor, ir al otro límite del actuador lineal.
- Si se introduce algo que **no** sea un **número**, mostraréis un mensaje de error en la consola y el actuador no hará nada.

**Vídeo**: para enseñar que funciona, grabad la pantalla y que se vea como introducís un conjunto de coordenadas y cómo después el actuador se mueve. Enseñad varios conjuntos de coordenadas, y en alguno de ellos introducid al menos una coordenada negativa y una mayor del límite existente. Hay que enseñar también las 3 excepciones.

## (Precisión4.7) Mejora de la precisión (0,1 Puntos)

*Material necesario: 1 actuador lineal y 2 sensores de colisión.*

Requiere tener implementado el ejercicio principal 4.4 [y tal vez también el 4.5]

Aumentar el rango de posiciones actuales [0-24] a otro significativamente mayor [0-N], dónde N se pueda especificar y pudiera ser 75, 100, 500, etc., es decir, una posición con la mayor precisión posible. La N debe de pedirla por consola. Tras esto, el programa deberá realizar los cálculos necesarios.

**Vídeo**:

- Grabad la pantalla y como introducís la N y la coordenada para ver cómo el actuador se mueve.
- Enseñad varias coordenadas.
- Probad con una coordenada negativa.
- Probad una coordenada mayor del límite existente.
- Probad a introducir caracteres (excepción creada en el 4.5).
- Probar varias N diferentes.