MANUAL COMPLETO DEL MINICURSO · CURSOSMECATRONICA.COM

Sistemas Robóticos

Tipos de robots, configuraciones y aplicaciones, sensores, control y seguridad, cinemática con Denavit–Hartenberg, exactitud y trayectorias, actuadores y dimensionamiento.

Cómo usar este manual. Cada capítulo corresponde a un tramo del video (se indica el minuto). Mira el tramo, estudia el manual y ejecuta el programa del capítulo: python archivo.py. Todos usan solo la biblioteca estándar de Python 3.8 o superior y reproducen las cifras de los ejemplos resueltos.

Contenido

N.º	Capítulo	Video
00	**Introducción** Qué cubre el minicurso y cómo usar el material.	min 0:00
01	**Tipos de robots y terminología del brazo robótico** Clasificación, robots serie y paralelos, AGV y AMR, articulaciones, grados de libertad, muñeca, TCP y carga útil.	min 0:51
02	**Configuraciones del brazo y aplicaciones industriales** Cartesiano, cilíndrico, polar, SCARA, articulado y delta; soldadura, pintura, paletizado y tiempo de ciclo.	min 15:49
03	**Sensores, control y seguridad en sistemas robóticos** Encoders y resolvers, fuerza y par, visión, controlador y EtherCAT, teach pendant, ISO 10218 y distancia de separación.	min 31:06
04	**Cinemática de manipuladores: Denavit–Hartenberg y jacobiano** Rotaciones, transformaciones homogéneas, parámetros DH, cinemática directa e inversa, jacobiano y singularidades.	min 45:56
05	**Exactitud, repetibilidad y planificación de trayectorias** Resolución, ISO 9283, fuentes de error, calibración del TCP, PTP y lineal, polinomios y perfiles trapezoidal y S.	min 1:01:05
06	**Actuadores, transmisiones armónicas y dimensionamiento** Actuadores eléctricos, neumáticos e hidráulicos, PMSM y FOC, lazos en cascada, reductores y par eficaz.	min 1:16:17
07	**Resumen y cierre** Síntesis del minicurso.	min 1:31:18

Curso gratuito en video: **cursosmecatronica.com/cursos/sistemas-roboticos/**

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Introducción

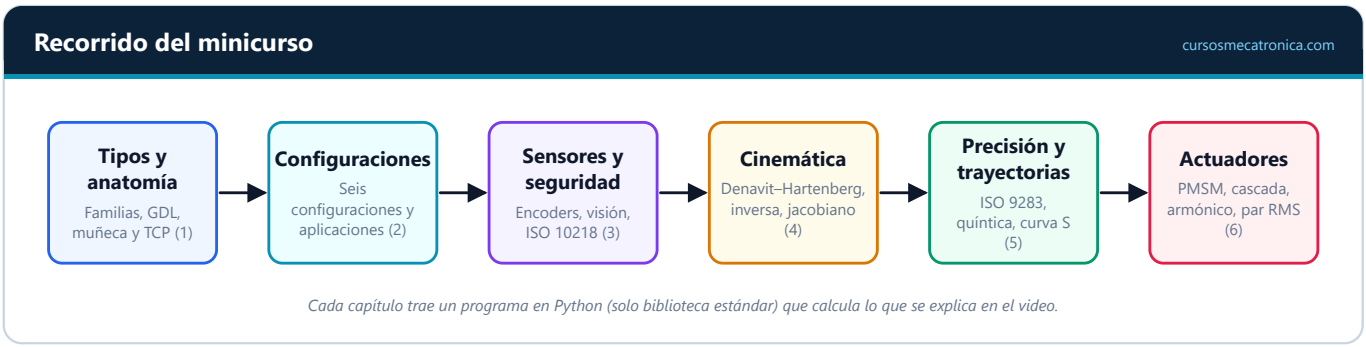
Introducción · En el video: **minuto 0:00** · Duración 0:51

Objetivo. Conocer el recorrido del minicurso y preparar el material de trabajo.

Ideas clave

De la anatomía al dimensionamiento

El minicurso parte con los tipos de robots y la terminología del brazo, sigue con las configuraciones y sus aplicaciones, y luego integra sensores, control y seguridad. La segunda mitad es cuantitativa: cinemática con Denavit–Hartenberg, exactitud y trayectorias, y el dimensionamiento de motores y reductores.



Estructura del minicurso.

Cómo estudiar

Cada capítulo coincide con un tramo del video. Mira el tramo, lee el manual y ejecuta el programa del capítulo con `python archivo.py`: todos usan solo la biblioteca estándar y muestran las mismas cifras de los ejemplos resueltos.

Autoevaluación

1. ¿Qué necesita el código del minicurso?

Respuestas. 1. Python 3.8 o superior; no usa bibliotecas externas.

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Tipos de robots y terminología del brazo robótico

Capítulo 01 · En el video: **minuto 0:51** · Duración 14:58

Objetivo. Clasificar un robot, contar sus grados de libertad y verificar que una herramienta no sobrecarga la muñeca.

Ideas clave

Familias de robots

ISO 8373 define el robot industrial como un manipulador multipropósito, reprogramable, con tres o más ejes. En planta conviven el robot industrial en celda cerrada, el robot colaborativo (cobot) con sensado de par, el robot paralelo delta para recoger y colocar a alta velocidad y el robot móvil autónomo (AMR).

Serie frente a paralelo

Un manipulador serie es una cadena abierta: cada eslabón carga con los siguientes y los errores se suman, pero el espacio de trabajo es grande. Un robot paralelo cierra la cadena con varias patas que sostienen la plataforma: es rígido y liviano en movimiento, con un espacio de trabajo pequeño.

De AGV a AMR

El AGV sigue una guía fija (cinta magnética o línea); el AMR se localiza con LiDAR y mapeo simultáneo (SLAM) y planifica su ruta, por lo que esquiva obstáculos y se reprograma sin obras.

Articulaciones y grados de libertad

Los pares inferiores más usados son el rotacional (R) y el prismático (P), con un grado de libertad cada uno. La movilidad se calcula con Grübler–Kutzbach: $M = 6(n - 1 - j) + \Sigma f_i$. Para ubicar y orientar una herramienta en el espacio se necesitan seis.



Cadena cinemática abierta de un robot 6R.

Muñeca, brida y TCP

En la muñeca esférica los tres últimos ejes se cortan en un punto, lo que separa el problema de posición del de orientación. La herramienta se monta en la brida normalizada ISO 9409 y el punto de trabajo que se programa es el TCP (tool center point).

Espacio de trabajo y carga útil

El espacio alcanzable reúne los puntos que el TCP puede tocar; el espacio diestro, los que alcanza con cualquier orientación. La carga útil nominal se da con su centro de gravedad: lo que limita la muñeca es el momento

$M = m \cdot g \cdot d$, no solo la masa.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Movilidad espacial	$M = 6(n - 1 - j) + \Sigma f_i$	n eslabones con la base, j juntas.
Movilidad plana	$M = 3(n - 1 - j) + \Sigma f_i$	Cuadrilátero articulado: M = 1.
Momento en la brida	$M = \Sigma\, m_k \cdot g \cdot d_k$	d: distancia del centro de gravedad a la brida.
Alcance de un brazo 2R	$A = \pi[(l_1+l_2)^2 - (l_1-l_2)^2]$	Corona entre el radio mínimo y el máximo.

Ejemplo resuelto: momento estático en la muñeca

1. Pieza de 8 kg con el centro de gravedad a 120 mm de la brida y garra de 2 kg a 60 mm.
2. $M = 9{,}81 \cdot (8 \cdot 0{,}120 + 2 \cdot 0{,}060) = 10{,}59$ N·m.
3. El catálogo admite 12 N·m en la muñeca: se usa el 88 %.
4. Distancia máxima del centro de gravedad de la pieza: $(12 - 2 \cdot 9{,}81 \cdot 0{,}06)/(8 \cdot 9{,}81) = 0{,}138$ m.

La herramienta cumple con poco margen: una pieza más larga obligaría a acortar la garra o a elegir un robot mayor (movilidad_y_carga.py).

Familias de robots

	Industrial	Colaborativo	Delta	AMR
Estructura	Serie 6R	Serie 6R/7R con sensores de par	Paralela	Base móvil
Fuerte en	Carga, velocidad, alcance	Trabajo junto a personas	Recoger y colocar muy rápido	Logística flexible
Seguridad	Celda con resguardos	Limitación de potencia y fuerza	Celda cerrada	Escáneres láser

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

movilidad_y_carga.py — movilidad de cinco mecanismos, momento en la muñeca y alcance de un brazo plano.

```
"""
movilidad_y_carga.py — Capítulo 01: grados de libertad, carga en la muñeca y alcance.

1) Criterio de Grübler-Kutzbach para varios mecanismos.
2) Momento estático en la brida de la muñeca frente al límite del catálogo.
3) Área del espacio alcanzable de un brazo plano de dos eslabones.
"""
import math

G = 9.81  # m/s²
```

```python
def grubler(eslabones, juntas, libertades, espacial=True):
    """Movilidad M = λ(n − 1 − j) + Σ f_i  (λ = 6 en el espacio, 3 en el plano)."""
    lam = 6 if espacial else 3
    return lam * (eslabones - 1 - juntas) + sum(libertades)


MECANISMOS = [
    # nombre, eslabones (con la base), juntas, libertades por junta, espacial
    ('Brazo articulado 6R', 7, 6, [1] * 6, True),
    ('SCARA RRPR', 5, 4, [1] * 4, True),
    ('Cuadrilátero articulado plano', 4, 4, [1] * 4, False),
    ('Cinco barras plano (2 motores)', 5, 5, [1] * 5, False),
    ('Pórtico cartesiano PPP', 4, 3, [1] * 3, True),
]


def momento_muneca(cargas):
    """cargas: lista de (masa kg, distancia del centro de gravedad a la brida en m)."""
    return sum(m * G * d for m, d in cargas)


def area_alcance_2r(l1, l2):
    """Corona circular entre |l1 − l2| y l1 + l2 (giro completo de ambas juntas)."""
    return math.pi * ((l1 + l2) ** 2 - (l1 - l2) ** 2)


if __name__ == '__main__':
    print('Movilidad (Grübler-Kutzbach)')
    for nombre, n, j, f, esp in MECANISMOS:
        print(f'  {nombre:<34s} M = {grubler(n, j, f, esp)}')

    cargas = [(8.0, 0.120), (2.0, 0.060)]   # pieza y garra
    m = momento_muneca(cargas)
    limite = 12.0                           # N·m admisibles en la muñeca (catálogo)
    print(f'\nMomento en la brida: {m:.2f} N·m  (límite {limite:.1f} N·m)'
          f' -> uso {100 * m / limite:.0f} %  {"OK" if m <= limite else "EXCEDE"}')
    # Distancia máxima del CG de la pieza para no superar el límite
    d_max = (limite - 2.0 * G * 0.060) / (8.0 * G)
    print(f'Distancia máxima del CG de la pieza de 8 kg: {1000 * d_max:.0f} mm')

    l1, l2 = 0.40, 0.30
    print(f'\nBrazo plano {l1:.2f} m + {l2:.2f} m: radio de {abs(l1 - l2):.2f} a {l1 + l2:.2f} m,'
          f' área alcanzable {area_alcance_2r(l1, l2):.3f} m²')
```

Errores frecuentes

- Mirar solo la masa de la carga y olvidar la distancia de su centro de gravedad.
- Contar mal los eslabones en Grübler (la base cuenta como eslabón).
- Confundir espacio alcanzable con espacio diestro.

Autoevaluación

1. ¿Cuántos grados de libertad hacen falta para ubicar y orientar una herramienta?
2. ¿Qué ventaja da la muñeca esférica?
3. ¿Qué movilidad tiene un cuadrilátero articulado plano?

Respuestas. 1. Seis. **2.** Separa la posición (J1–J3) de la orientación (J4–J6). **3.** M = 1.

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Configuraciones del brazo y aplicaciones industriales

Capítulo 02 · En el video: **minuto 15:49** · Duración 15:17

Objetivo. Elegir la configuración de un robot según su espacio de trabajo y comprobar que su ciclo cabe en el takt de la planta.

Ideas clave

Cartesiano y pórtico (PPP)

Tres ejes prismáticos perpendiculares: el espacio de trabajo es un prisma, la rigidez es uniforme y la programación es directa. Los pórticos (gantry) cubren áreas grandes sobre la línea.

Cilíndrico (RPP) y polar (RRP)

Una base rotacional con brazo que sube y se extiende genera un cilindro hueco; el polar cambia la columna por un giro de elevación y su envolvente es un casquete esférico.

SCARA

Dos brazos rotacionales de ejes verticales y un eje Z: es rígido en vertical y flexible en el plano horizontal, ideal para insertar piezas con pequeños desalineamientos.

Articulado (RRR) y delta

El brazo antropomórfico tiene el mayor alcance para su tamaño y llega a orientaciones difíciles. El delta mueve una plataforma liviana con tres brazos paralelos y supera las 100 recogidas por minuto.

Aplicaciones

Soldadura por arco (trayectoria continua y velocidad constante) y por puntos (pinza servoneumática); pintura (muñeca hueca y protección para atmósferas explosivas); paletizado con garras de vacío Venturi y sincronía con la cinta por encoder; desbarbado, que exige rigidez frente a las fuerzas de corte.

Tiempo de ciclo y takt

El takt es el ritmo que pide la planta: tiempo disponible dividido por la demanda. El ciclo del robot (aproximar, procesar, retirar y trasladar) debe quedar por debajo, con margen para paradas.



Desglose del ciclo de configuraciones_y_takt.py.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Volumen cartesiano	$V = x \cdot y \cdot z$	Prisma recto.
Volumen cilíndrico	$V = \pi(r_2^{\,2} - r_1^{\,2}) \cdot h \cdot (\varphi/360°)$	φ: giro de la base.
Volumen esférico	$V = (r_2^{\,3} - r_1^{\,3})/3 \cdot (\text{sen } e_{máx} - \text{sen } e_{mín}) \cdot \varphi$	φ en radianes; e: elevación.
Takt	$T_{takt} = \text{tiempo disponible} / \text{demanda}$	Ritmo de la línea.
Intercepción en cinta	$t = (x_{robot} - x_{pieza}) / v_{cinta}$	Cuándo llega la pieza.

Ejemplo resuelto: una celda de soldadura frente al takt

1. Turno de 8 h con 30 min de pausas: 27 000 s disponibles; demanda 1800 piezas.
2. $T_{takt} = 27\,000/1800 = 15{,}0$ s.
3. Ciclo: 1,2 + 9,5 + 1,0 + 2,1 = 13,8 s.
4. Ocupación: 13,8/15 = 92 %.

La celda cumple, pero con 1,2 s de margen por pieza. configuraciones_y_takt.py también calcula volúmenes de trabajo e intercepción en cinta.

Selección de la configuración

Configuración	Espacio de trabajo	Fuerte en	Uso típico
Cartesiano PPP	Prisma	Rigidez, áreas grandes	Pórticos, CNC, dispensado
Cilíndrico RPP	Cilindro hueco	Carga y descarga radial	Máquinas herramienta
Polar RRP	Casquete esférico	Alcance	Fundición, forja
SCARA	Cilindro	Velocidad en el plano	Ensamble electrónico
Articulado RRR	Esfera irregular	Alcance y orientación	Soldadura, pintura, manipulación
Delta	Cúpula pequeña	Velocidad muy alta	Envasado, alimentos

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

configuraciones_y_takt.py — volúmenes de trabajo, takt, ocupación e intercepción de una pieza en la cinta.

```
"""
configuraciones_y_takt.py — Capítulo 02: volumen de trabajo, tiempo de ciclo e intercepción en cinta.
"""
import math


def vol_cartesiano(x, y, z):
    return x * y * z


def vol_cilindrico(r_min, r_max, h, giro_grados=360):
```

```
    return math.pi * (r_max ** 2 - r_min ** 2) * h * giro_grados / 360


def vol_esferico(r_min, r_max, giro_grados=360, elev_min=-30, elev_max=60):
    """Casquete esférico entre dos radios y dos ángulos de elevación."""
    franja = math.sin(math.radians(elev_max)) - math.sin(math.radians(elev_min))
    return (2 * math.pi / 3) * (r_max ** 3 - r_min ** 3) * franja * giro_grados / 360


def takt(tiempo_disponible_s, demanda):
    return tiempo_disponible_s / demanda


def intercepcion(x_pieza, v_cinta, x_robot, t_robot):
    """Instante en que la pieza llega a x_robot y si el robot alcanza a estar listo."""
    t_llegada = (x_robot - x_pieza) / v_cinta
    return t_llegada, t_llegada >= t_robot


if __name__ == '__main__':
    print('Volumen de trabajo aproximado (m³)')
    print(f'  Cartesiano 1,2 × 0,8 × 0,5 m          : {vol_cartesiano(1.2, 0.8, 0.5):.3f}')
    print(f'  Cilíndrico r 0,3–0,9 m, h 0,6 m, 300° : {vol_cilindrico(0.3, 0.9, 0.6, 300):.3f}')
    print(f'  Esférico r 0,4–1,2 m, 340°, −30/+60°  : {vol_esferico(0.4, 1.2, 340):.3f}')

    disponible = 8 * 3600 - 30 * 60          # turno menos pausas
    t_takt = takt(disponible, 1800)
    etapas = {'aproximación': 1.2, 'proceso': 9.5, 'retirada': 1.0, 'traslado': 2.1}
    ciclo = sum(etapas.values())
    print(f'\nTakt = {disponible} s / 1800 piezas = {t_takt:.1f} s')
    print(f'Ciclo del robot = {ciclo:.1f} s -> ocupación {100 * ciclo / t_takt:.0f} %'
          f' {"(cumple)" if ciclo <= t_takt else "(NO cumple)"}')

    t, ok = intercepcion(x_pieza=0.0, v_cinta=0.25, x_robot=0.80, t_robot=2.6)
    print(f'\nPieza a 0,25 m/s llega a la zona del robot en {t:.2f} s;'
          f' el robot necesita 2,6 s -> {"alcanza" if ok else "no alcanza"}')
```

Errores frecuentes

- Calcular el takt con el turno completo sin descontar pausas.
- Elegir un SCARA para tareas que necesitan inclinar la herramienta.
- Olvidar el tiempo de aproximación y retirada en el ciclo.

Autoevaluación

1. ¿Qué configuración tiene rigidez vertical alta y conformidad horizontal?
2. Si el takt es 15 s y el ciclo 13,8 s, ¿qué ocupación tiene el robot?
3. ¿Por qué los robots de pintura usan muñeca hueca?

Respuestas. 1. SCARA. **2.** 92 %. **3.** Para llevar las mangueras por dentro y evitar enredos y chispas.

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Sensores, control y seguridad en sistemas robóticos

Capítulo 03 · En el video: **minuto 31:06** · Duración 14:50

Objetivo. Relacionar los sensores y el controlador de un robot con las normas de seguridad y calcular la distancia mínima de un escáner.

Ideas clave

Cinco subsistemas

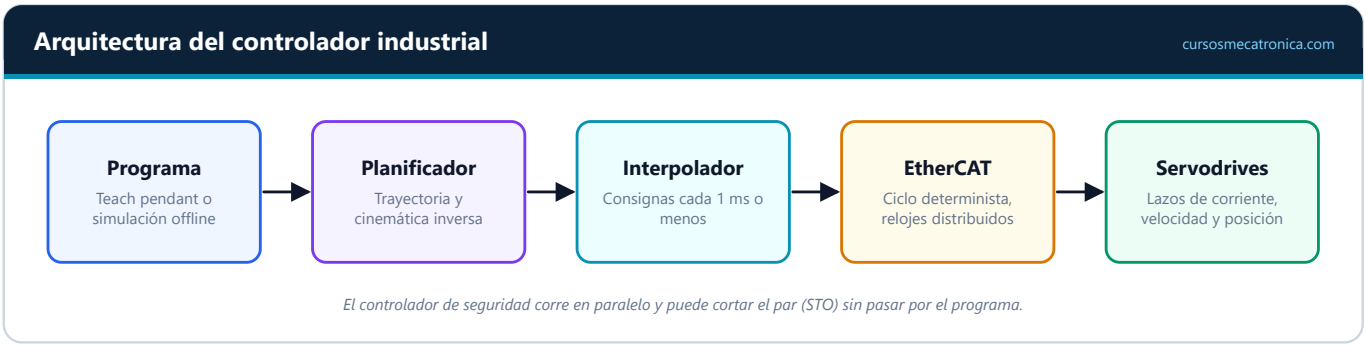
Estructura mecánica, actuación, sensórica, control y seguridad. Las aleaciones livianas y el arnés interno guiado reducen la inercia y alargan la vida de los cables que se tuercen en cada ciclo.

Sensores propioceptivos

Miden el estado del propio robot. El encoder incremental en cuadratura entrega 4 cuentas por línea; el multivuelta absoluto no necesita referenciarse; el resolver es robusto ante temperatura y vibración. Con el encoder en el motor, la resolución en la articulación se multiplica por la reducción.

Sensores exteroceptivos

Miden el entorno. El sensor de fuerza y par de seis ejes en la muñeca (galgas extensiométricas) permite controlar el contacto al pulir o ensamblar. La visión puede ir en la muñeca (eye-in-hand) o fija en la celda; el LiDAR y las cámaras 3D entregan nubes de puntos.



Del programa al motor.

Teach pendant

La consola trae parada de emergencia y un dispositivo habilitador de tres posiciones: solo permite mover el robot con el botón a medio camino; si se suelta o se aprieta con fuerza (pánico), se detiene. La programación offline simula la celda antes de llevarla al robot.

ISO 10218 e ISO/TS 15066

Los cuatro modos colaborativos son parada monitoreada, guiado manual, monitoreo de velocidad y separación, y limitación de potencia y fuerza. Este último compara el par externo estimado con umbrales biomecánicos.



Salida de seguridad_celda.py.

Distancia de separación

ISO 13855 fija dónde poner un escáner o cortina: $S = K \cdot T + C$, con K = 1600 mm/s (velocidad de aproximación de una persona) y T el tiempo del detector más el de parada del robot.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Resolución articular	$\Delta\theta = 360° / (4 \cdot N \cdot i)$	N líneas del encoder, i reducción.
Resolución en el TCP	$\Delta s = R \cdot \Delta\theta$ (rad)	R: distancia al eje.
Distancia mínima (ISO 13855)	$S = K \cdot (t_1 + t_2) + C$	K = 1600 mm/s.
Escáner horizontal	$C = 1200 - 0{,}4 \cdot H$ (≥ 850 mm)	H: altura del plano de detección.
Par externo	$\tau_{ext} = \tau_{medido} - \tau_{modelo}$	Se compara con un umbral.

Ejemplo resuelto: dónde poner el escáner de una celda

1. Escáner con plano a 300 mm del piso: $C = 1200 - 0{,}4 \cdot 300 = 1080$ mm.
2. Tiempo del escáner 80 ms y tiempo de parada del robot 420 ms: T = 0,50 s.
3. $S = 1600 \cdot 0{,}50 + 1080 = 1880$ mm.
4. El campo de protección debe empezar a 1,88 m del punto peligroso más cercano.

Si el robot frenara en 300 ms, S bajaría a 1,68 m. seguridad_celda.py calcula además la resolución del encoder y simula la detección de un choque.

Modos colaborativos (ISO 10218 / ISO/TS 15066)

Modo	Cómo funciona	Ejemplo
Parada monitoreada	El robot se detiene cuando entra la persona	Carga manual de una pieza
Guiado manual	La persona mueve el robot con un mando en la muñeca	Enseñar posiciones
Velocidad y separación	Baja la velocidad según la distancia medida	Celda con escáner
Potencia y fuerza	Limita el par y la energía de contacto	Cobot junto al operario

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

seguridad_celda.py — resolución del encoder en el TCP, distancia ISO 13855 y supervisión del par externo.

```
"""
seguridad_celda.py — Capítulo 03: resolución del encoder, distancia de separación
(ISO 13855) y supervisión del par externo con parada de categoría 1.
"""
import math


def resolucion_tcp(pulsos_vuelta, reduccion, radio_m):
    """Paso angular en la articulación y desplazamiento en el TCP por cuenta (cuadratura ×4)."""
    cuentas = 4 * pulsos_vuelta * reduccion
    paso = 2 * math.pi / cuentas
    return math.degrees(paso), paso * radio_m


def distancia_iso13855(t_detector, t_parada, altura_mm, k=1600):
    """Escáner de plano horizontal: S = K·T + C, con C = 1200 − 0,4·H (mínimo 850 mm)."""
    c = max(850, 1200 - 0.4 * altura_mm)
    return k * (t_detector + t_parada) + c, c


def par_modelo(t):
    """Par que el modelo dinámico predice para el movimiento (N·m)."""
    return 30 + 8 * math.sin(2 * math.pi * 0.5 * t)


def par_medido(t):
    """Par estimado con la corriente del motor: modelo + ruido + choque a los 1,20 s."""
    ruido = 1.2 * math.sin(37 * t) + 0.8 * math.sin(91 * t)
    choque = 0.0 if t < 1.20 else min(60.0, 150 * (t - 1.20))
    return par_modelo(t) + ruido + choque


def supervisar(umbral=12.0, dt=0.001, duracion=2.0):
    """Devuelve el instante de detección y el de reposo (parada de categoría 1, 0,15 s)."""
    t = 0.0
    while t < duracion:
        externo = par_medido(t) - par_modelo(t)
        if abs(externo) > umbral:
            return t, t + 0.15
        t += dt
    return None, None


if __name__ == '__main__':
    grados, metros = resolucion_tcp(pulsos_vuelta=2500, reduccion=100, radio_m=1.2)
    print(f'Encoder 2500 ppr ×4, reductor 100:1 -> {grados * 3600:.2f} arcsec por cuenta,'
```

```
          f' {metros * 1e6:.2f} µm en el TCP a 1,2 m')

    s, c = distancia_iso13855(t_detector=0.080, t_parada=0.420, altura_mm=300)
    print(f'ISO 13855: C = {c:.0f} mm, S = 1600 × 0,50 + {c:.0f} = {s:.0f} mm')

    t_det, t_reposo = supervisar()
    print(f'Choque a 1,200 s -> detectado a {t_det:.3f} s'
          f' (umbral 12 N·m), robot en reposo a {t_reposo:.3f} s')
```

Errores frecuentes

- Usar el tiempo de parada del catálogo con carga mínima en vez de con la carga real.
- Olvidar el factor 4 de la cuadratura al calcular la resolución.
- Poner el escáner a la distancia S medida desde el robot y no desde el punto peligroso.

Autoevaluación

1. ¿Qué hace el habilitador si el operario lo aprieta a fondo?
2. ¿Qué valor de K usa ISO 13855 para una persona caminando?
3. ¿Qué mide un sensor exteroceptivo?

Respuestas. 1. Detiene el robot (posición de pánico). **2.** 1600 mm/s. **3.** El entorno (fuerza de contacto, imagen, distancia).

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Cinemática de manipuladores: Denavit–Hartenberg y jacobiano

Capítulo 04 · En el video: **minuto 45:56** · Duración 15:09

Objetivo. Calcular la pose del TCP a partir de los ángulos articulares, resolver el problema inverso y reconocer una singularidad.

Ideas clave

Pose y rotaciones

La pose del TCP son seis números: posición (x, y, z) y orientación. La orientación se guarda en una matriz de rotación R de 3×3: sus columnas son vectores unitarios perpendiculares, $R^T \cdot R = I$ y det R = 1, así que la inversa es la transpuesta.

Transformaciones homogéneas

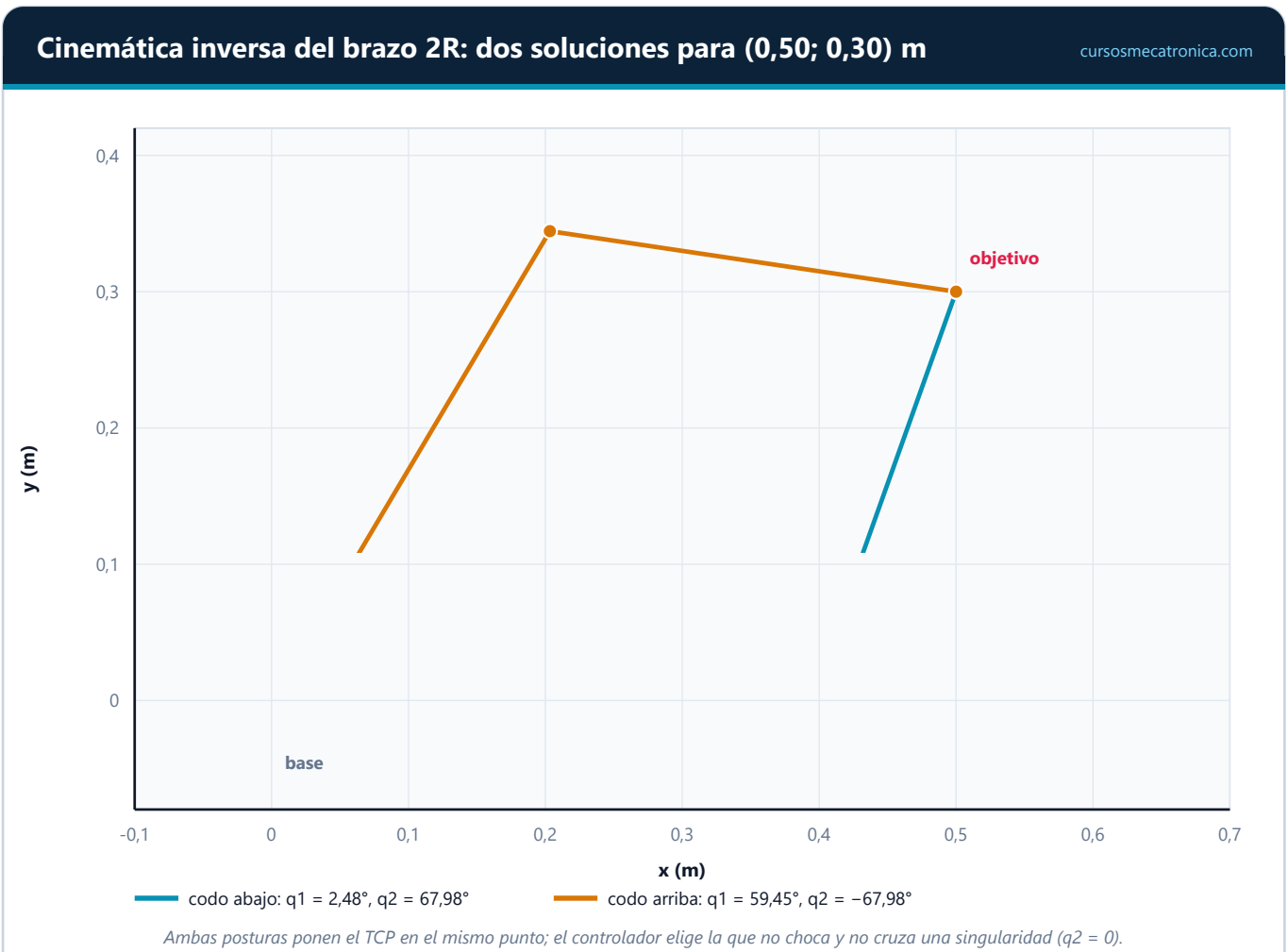
Una matriz 4×4 junta rotación y traslación: $T = [R\ p;\ 0\ 1]$. Encadenar marcos es multiplicar matrices: ${}^0T_n = {}^0T_1 \cdot {}^1T_2 \cdots$.

Parámetros de Denavit–Hartenberg

Cada eslabón se describe con cuatro números: θ (giro alrededor de z), d (desplazamiento a lo largo de z), a (largo de la normal común) y α (torsión entre ejes). En una junta rotacional θ es la variable; en una prismática, d.

Cinemática directa e inversa

La directa multiplica las matrices A_i y siempre tiene una solución. La inversa busca los ángulos para una pose dada: puede tener varias (codo arriba y codo abajo) o ninguna si el punto está fuera del alcance. Con muñeca esférica, el criterio de Pieper permite resolver primero la posición del centro de la muñeca y luego la orientación (ángulos de Euler ZYZ).

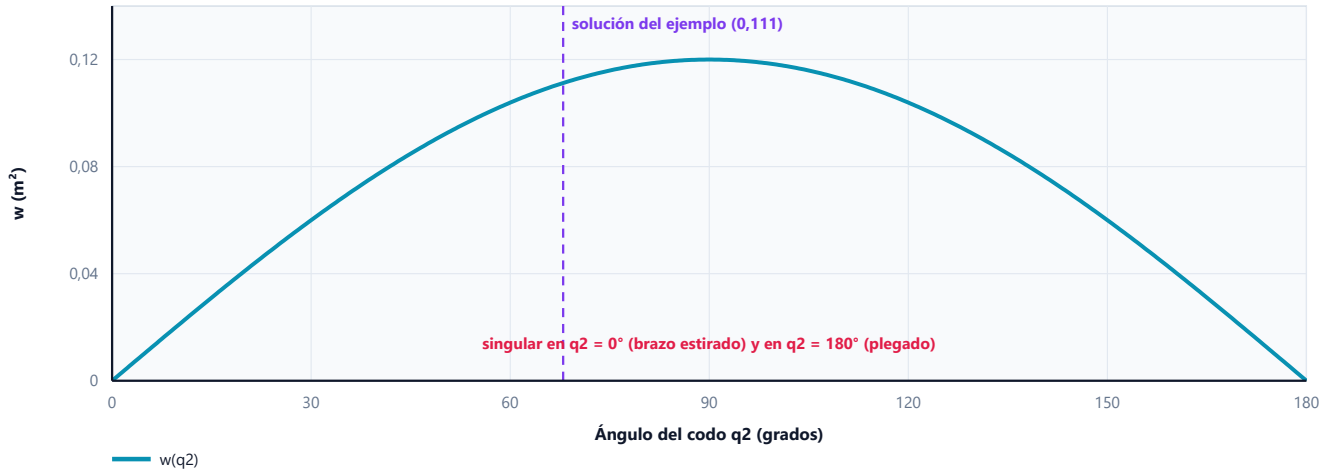


Salida de cinematica_dh.py.

Jacobiano y singularidades

El jacobiano relaciona velocidades: $v = J(q)\cdot\dot{q}$; su transpuesta lleva fuerzas del TCP a pares articulares ($\tau = J^{T}\cdot F$). Si det J = 0 el robot está en una singularidad: pierde una dirección de movimiento. El índice de Yoshikawa $w = \sqrt{\det(J\cdot J^{T})}$ mide cuán lejos está.

Manipulabilidad del brazo 2R: w = l1·l2·|sen q2| cursosmecatronica.com



Con w = 0 el jacobiano no tiene inversa: el TCP no puede moverse en alguna dirección y las velocidades articulares se disparan.

Índice de Yoshikawa del capítulo 04.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Matriz DH	$A_i = Rot_z(\theta)\cdot Tras_z(d)\cdot Tras_x(a)\cdot Rot_x(\alpha)$	Convención estándar.
Directa 2R	$x = l_1c_1 + l_2c_{12},\ y = l_1s_1 + l_2s_{12}$	c_{12} = cos(q_1+q_2).
Inversa 2R	$c_2 = (x^2 + y^2 - {l_1}^2 - {l_2}^2)/(2l_1l_2)$	Ley de cosenos; s_2 = ±√(1 − c_2^2).
Hombro	$q_1 = \text{atan2}(y, x) - \text{atan2}(l_2s_2, l_1 + l_2c_2)$	Usar atan2, no atan.
Jacobiano 2R	$\det J = l_1\cdot l_2\cdot \text{sen}\ q_2$	Singular con el brazo estirado o plegado.

Ejemplo resuelto: inversa del brazo 2R hacia (0,50; 0,30) m

1. l_1 = 0,40 m, l_2 = 0,30 m; $x^2 + y^2$ = 0,34.
2. $c_2 = (0{,}34 - 0{,}16 - 0{,}09)/0{,}24 = 0{,}375 \rightarrow q_2$ = ±67,98°.
3. Codo abajo: $q_1 = 30{,}96° - 28{,}48° = 2{,}48°$. Codo arriba: q_1 = 59,45°.
4. Comprobación con la directa: ambas dan (0,5000; 0,3000). det J = ±0,1112 m².

Dos posturas válidas, lejos de la singularidad (w máx = 0,12 m² con q_2 = 90°). cinematica_dh.py hace el cálculo con matrices DH.

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

cinematica_dh.py — matrices DH, cinemática directa e inversa 2R, jacobiano, manipulabilidad y velocidad del TCP. *(extracto: primeras 60 de 70 líneas)*

```
"""
cinematica_dh.py — Capítulo 04: transformaciones homogéneas, Denavit-Hartenberg,
```

```
cinemática directa e inversa de un brazo plano 2R, jacobiano y manipulabilidad.
Sin NumPy: matrices como listas para ver cada operación.
"""
import math


def matmul(a, b):
    return [[sum(a[i][k] * b[k][j] for k in range(4)) for j in range(4)] for i in range(4)]


def dh(theta, d, a, alfa):
    """A_i = Rot_z(θ)·Trans_z(d)·Trans_x(a)·Rot_x(α)  (convención estándar)."""
    ct, st, ca, sa = math.cos(theta), math.sin(theta), math.cos(alfa), math.sin(alfa)
    return [[ct, -st * ca, st * sa, a * ct],
            [st, ct * ca, -ct * sa, a * st],
            [0.0, sa, ca, d],
            [0.0, 0.0, 0.0, 1.0]]


def directa(tabla, q):
    """tabla: filas (d, a, alfa) de juntas rotacionales; q: ángulos en rad."""
    t = [[1.0 if i == j else 0.0 for j in range(4)] for i in range(4)]
    for (d, a, alfa), qi in zip(tabla, q):
        t = matmul(t, dh(qi, d, a, alfa))
    return t


def inversa_2r(x, y, l1, l2, codo_arriba=False):
    c2 = (x * x + y * y - l1 * l1 - l2 * l2) / (2 * l1 * l2)
    if abs(c2) > 1:
        raise ValueError('punto fuera del alcance')
    s2 = math.sqrt(1 - c2 * c2) * (-1 if codo_arriba else 1)
    q2 = math.atan2(s2, c2)
    q1 = math.atan2(y, x) - math.atan2(l2 * s2, l1 + l2 * c2)
    return q1, q2


def jacobiano_2r(q1, q2, l1, l2):
    s1, c1 = math.sin(q1), math.cos(q1)
    s12, c12 = math.sin(q1 + q2), math.cos(q1 + q2)
    return [[-l1 * s1 - l2 * s12, -l2 * s12],
            [l1 * c1 + l2 * c12, l2 * c12]]


if __name__ == '__main__':
    l1, l2 = 0.40, 0.30
    tabla = [(0.0, l1, 0.0), (0.0, l2, 0.0)]
    x, y = 0.50, 0.30
    print(f'Objetivo ({x:.2f}, {y:.2f}) m con l1 = {l1} m y l2 = {l2} m')
    for arriba in (False, True):
        q1, q2 = inversa_2r(x, y, l1, l2, arriba)
        t = directa(tabla, (q1, q2))
        j = jacobiano_2r(q1, q2, l1, l2)
        det = j[0][0] * j[1][1] - j[0][1] * j[1][0]
        nombre = 'codo arriba' if arriba else 'codo abajo '
        print(f'  {nombre}: q1 = {math.degrees(q1):7.2f}°, q2 = {math.degrees(q2):7.2f}°'
              f' -> directa ({t[0][3]:.4f}, {t[1][3]:.4f})  det J = {det:+.4f}')
```

Errores frecuentes

- Usar atan en vez de atan2 y caer en el cuadrante equivocado.
- Mezclar grados y radianes en las funciones trigonométricas.
- Multiplicar las matrices en el orden inverso (el producto no es conmutativo).

- Pedir un punto fuera del alcance: $|c_2| > 1$.

Autoevaluación

1. ¿Cuál es la inversa de una matriz de rotación?
2. ¿Qué parámetro DH es variable en una junta rotacional?
3. ¿Cuándo es singular el brazo 2R?

Respuestas. 1. Su transpuesta. **2.** θ. **3.** Cuando sen $q_2 = 0$: estirado o plegado.

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Exactitud, repetibilidad y planificación de trayectorias

Capítulo 05 · En el video: **minuto 1:01:05** · Duración 15:12

Objetivo. Medir la exactitud y la repetibilidad de un robot y generar trayectorias suaves para sus articulaciones.

Ideas clave

Resolución, exactitud y repetibilidad

La resolución es el menor paso que el control puede ordenar. La exactitud (AP) es la distancia entre la pose ordenada y el promedio de las alcanzadas; la repetibilidad (RP) es el radio de la nube de llegadas: $\mathrm{RP} = \bar{l} + 3 \cdot S_l$.

Un robot suele repetir mucho mejor de lo que acierta.

Fuentes de error

Flexión elástica de eslabones y reductores bajo carga, juego (holgura), histéresis, dilatación térmica y errores de los parámetros geométricos. La calibración corrige los sistemáticos; los aleatorios fijan la repetibilidad.

Marcos y calibración del TCP

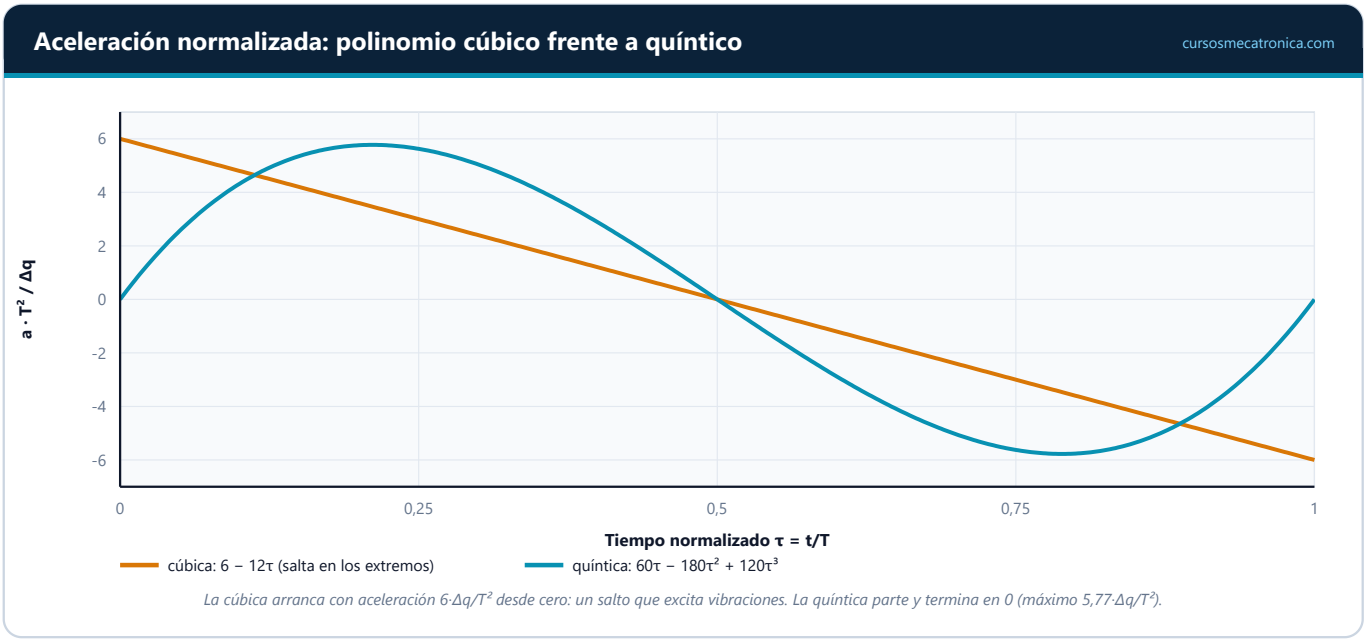
Mundo, base del robot, herramienta y objeto de trabajo. El método de cuatro puntos lleva la punta de la herramienta al mismo punto con cuatro orientaciones distintas y resuelve dónde está el TCP respecto de la brida.

PTP frente a trayectoria continua

En punto a punto cada eje llega cuando puede y el camino del TCP no está definido: es lo más rápido. En movimiento lineal o circular el TCP sigue una recta o un arco: hace falta para soldar o sellar.

Polinomios y perfiles

Un cúbico fija posición y velocidad en los extremos, pero la aceleración salta. El quíntico agrega aceleración nula al inicio y al final. En la industria se usan perfiles trapezoidales (aceleración, crucero, frenado) y curvas S, que limitan la sacudida (jerk) y evitan vibraciones.



Comparación de interpoladores del capítulo 05.

Rutas libres de colisión

Los obstáculos se representan en el espacio de configuraciones; se buscan caminos con campos de potencial, búsqueda en grafos o muestreo aleatorio (RRT*).

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Exactitud de posición	$\mathrm{AP} = \lVert\, \bar{p} - p_{\mathrm{ordenada}} \,\rVert$	$\bar{p}$: baricentro de las llegadas.
Repetibilidad	$\mathrm{RP} = \bar{l} + 3{\cdot}S_l$	l: distancias al baricentro.
Quíntica	$q = q_0 + \Delta q{\cdot}(10\tau^3 - 15\tau^4 + 6\tau^5)$	τ = t/T.
Máximos de la quíntica	$v_{máx} = 1{,}875{\cdot}\Delta q/T;\ a_{máx} = 5{,}774{\cdot}\Delta q/T^2$	v en τ = 0,5.
Trapezoidal	$T = \Delta q/v + v/a\ (\text{si } \Delta q \geq v^2/a)$	Si no, T = 2√(Δq/a).

Ejemplo resuelto: ensayo ISO 9283 con 10 llegadas

1. Pose ordenada (800; 0; 600) mm; diez llegadas medidas con láser tracker.
2. Baricentro: (800,217; 0,131; 599,931) mm.
3. $\mathrm{AP} = \sqrt{(0{,}217^2 + 0{,}131^2 + 0{,}069^2)} = 0{,}263$ mm.
4. Distancias al baricentro: media más tres desviaciones → $\mathrm{RP} = 0{,}067$ mm.

El robot repite a ±0,07 mm pero acierta a 0,26 mm: conviene programar por enseñanza o calibrar. iso9283_y_trayectorias.py también genera una quíntica de 0 a 90° en 1,5 s (v máx 112,5 °/s, a máx 230,9 °/s²).

Perfiles de movimiento

Perfil	Continuo en	Ventaja	Inconveniente
Cúbico	Posición y velocidad	Simple	Saltos de aceleración
Quíntico	Hasta la aceleración	Suave	Pico de velocidad 1,875·Δq/T
Trapezoidal	Velocidad	Tiempo mínimo con v y a limitadas	Sacudida infinita en los quiebres
Curva S	Aceleración (jerk limitado)	Sin vibraciones	Algo más lento

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

iso9283_y_trayectorias.py — exactitud y repetibilidad ISO 9283, quíntica a 1 kHz y tiempo de un perfil trapezoidal.

```
"""
iso9283_y_trayectorias.py — Capítulo 05: exactitud y repetibilidad de posición (ISO 9283),
interpolación quíntica a 1 kHz y perfil trapezoidal.
"""
import math

# Pose ordenada y 10 llegadas medidas con un láser tracker (mm)
ORDENADA = (800.0, 0.0, 600.0)
MEDIDAS = [
    (800.21, 0.12, 599.93), (800.24, 0.15, 599.95), (800.19, 0.10, 599.91),
    (800.23, 0.13, 599.94), (800.20, 0.16, 599.92), (800.25, 0.11, 599.96),
    (800.22, 0.14, 599.90), (800.18, 0.12, 599.93), (800.24, 0.13, 599.95),
    (800.21, 0.15, 599.92),
]


def iso9283(ordenada, medidas):
    n = len(medidas)
    media = [sum(p[k] for p in medidas) / n for k in range(3)]
    ap = math.dist(media, ordenada)
    l = [math.dist(p, media) for p in medidas]
    lm = sum(l) / n
    s = math.sqrt(sum((x - lm) ** 2 for x in l) / (n - 1))
    return ap, lm + 3 * s, media


def quintica(q0, qf, T, t):
    """Reposo a reposo: posición, velocidad y aceleración (τ = t/T)."""
    d, tau = qf - q0, t / T
    q = q0 + d * (10 * tau ** 3 - 15 * tau ** 4 + 6 * tau ** 5)
    v = d / T * (30 * tau ** 2 - 60 * tau ** 3 + 30 * tau ** 4)
    a = d / T ** 2 * (60 * tau - 180 * tau ** 2 + 120 * tau ** 3)
    return q, v, a


def tiempo_trapezoidal(dist, vmax, amax):
    """Devuelve el tiempo total y si alcanza la velocidad de crucero."""
    if dist >= vmax ** 2 / amax:
        return dist / vmax + vmax / amax, True
    return 2 * math.sqrt(dist / amax), False


if __name__ == '__main__':
    ap, rp, media = iso9283(ORDENADA, MEDIDAS)
    print(f'Baricentro: ({media[0]:.3f}, {media[1]:.3f}, {media[2]:.3f}) mm')
    print(f'Exactitud AP = {ap:.3f} mm   Repetibilidad RP = {rp:.3f} mm')
```

```
q0, qf, T = 0.0, 90.0, 1.5               # grados, segundos
muestras = [quintica(q0, qf, T, k / 1000) for k in range(int(T * 1000) + 1)]
vmax = max(abs(m[1]) for m in muestras)
amax = max(abs(m[2]) for m in muestras)
print(f'\nQuíntica 0° -> 90° en 1,5 s ({len(muestras)} consignas a 1 kHz):')
print(f'  v máx = {vmax:.2f} °/s (teórico 1,875·Δ/T = {1.875 * 90 / 1.5:.2f})')
print(f'  a máx = {amax:.2f} °/s² (teórico 5,774·Δ/T² = {5.7735 * 90 / 1.5 ** 2:.2f})')
print(f'  q(0,75 s) = {quintica(q0, qf, T, 0.75)[0]:.2f}°')

for dist in (90.0, 20.0):
    t, crucero = tiempo_trapezoidal(dist, vmax=120.0, amax=400.0)
    print(f'Trapezoidal {dist:.0f}° (120 °/s, 400 °/s²): {t:.3f} s'
          f' {"con" if crucero else "sin"} tramo de crucero')
```

Errores frecuentes

- Confundir exactitud con repetibilidad.
- Generar trayectorias con aceleración discontinua y culpar después al lazo de control por las vibraciones.
- Usar PTP donde el TCP debe seguir una línea.

Autoevaluación

1. ¿Cuál suele ser mejor en un robot, la exactitud o la repetibilidad?
2. ¿Qué condiciones agrega la quíntica a la cúbica?
3. ¿Cuánto tarda un trapezoidal de 20° con 120 °/s y 400 °/s^2?

Respuestas. 1. La repetibilidad. **2.** Aceleración nula al inicio y al final. **3.** 0,447 s, sin tramo de crucero.

MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Actuadores, transmisiones armónicas y dimensionamiento

Capítulo 06 · En el video: **minuto 1:16:17** · Duración 15:01

Objetivo. Elegir el actuador y el reductor de una articulación y verificar el motor por par pico y par eficaz.

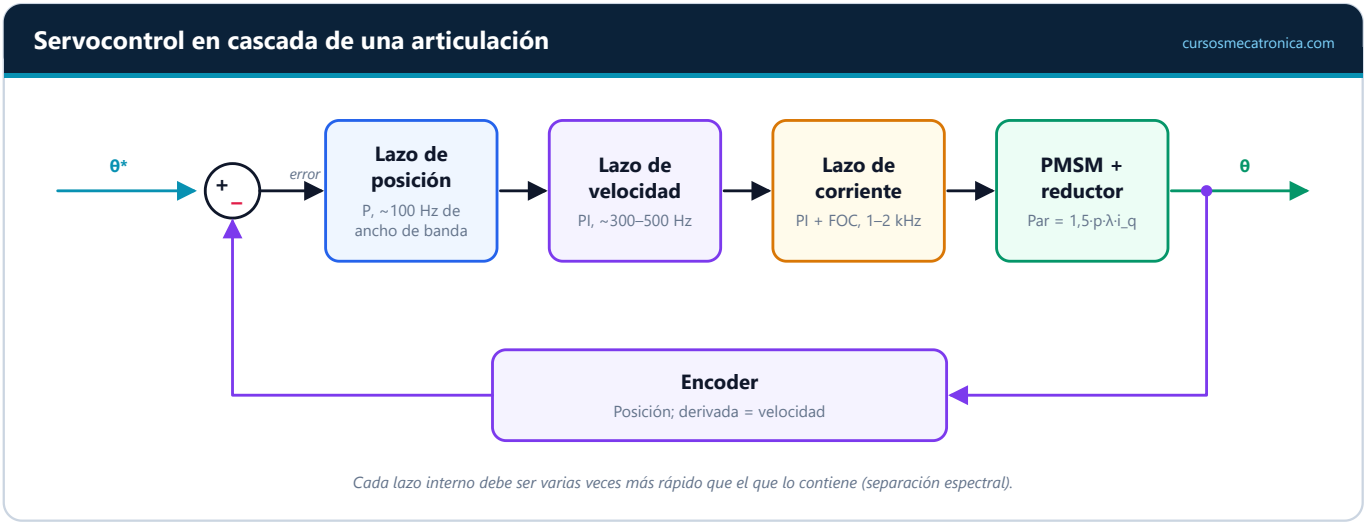
Ideas clave

Tres familias de actuadores

El servomotor eléctrico domina por precisión, limpieza y control; la neumática es barata y rápida en movimientos de todo o nada (garras); la hidráulica entrega la mayor fuerza por volumen en robots pesados.

PMSM y control de campo orientado

El servomotor síncrono de imanes de neodimio se controla con FOC: las transformadas de Clarke y Park pasan las corrientes de fase a un marco que gira con el rotor, donde i_q produce par y i_d se mantiene en cero.



Lazos de corriente, velocidad y posición.

Reductores sin juego

El reductor armónico deforma una corona flexible con un generador elíptico: con 200 dientes contra 202 logra 100:1 en una etapa y sin holgura. El cicloidal (RV) es más rígido y soporta choques: va en las primeras articulaciones de robots grandes.

Inercia reflejada

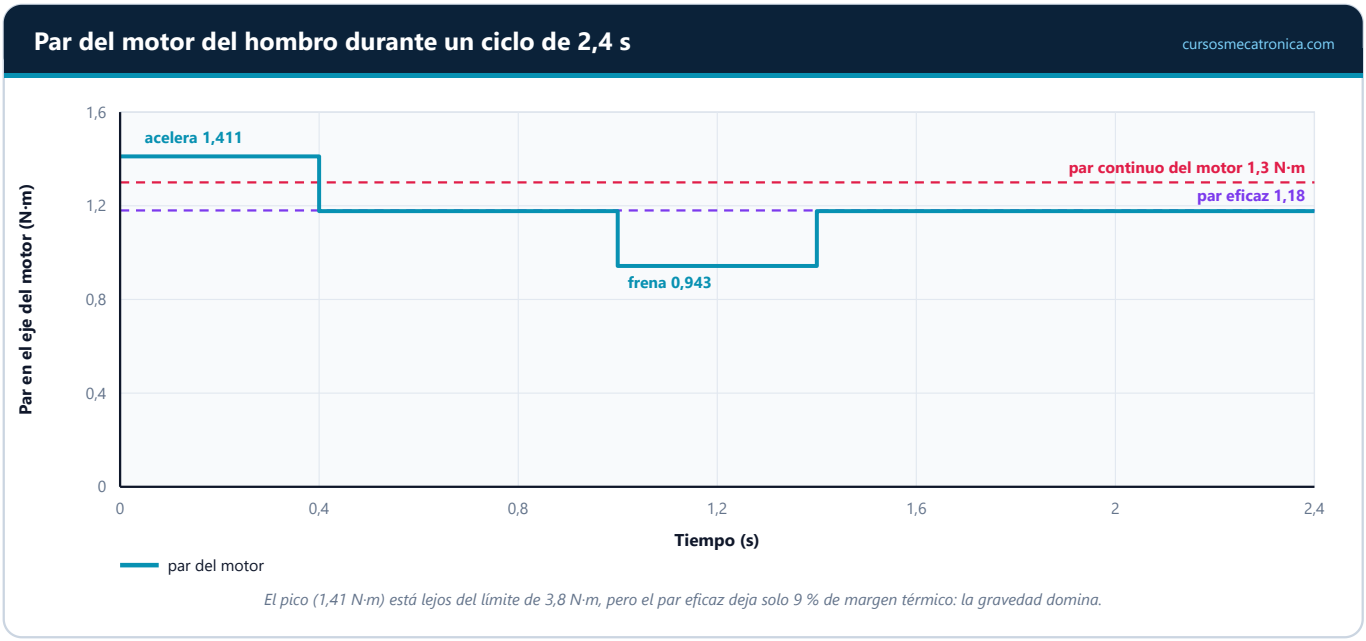
A través de una reducción i, la inercia de la carga se ve en el motor dividida por i^2. La relación que reparte la inercia por igual es $i_{ópt} = \sqrt{(J_{carga}/J_{motor})}$.

Actuación colaborativa

Los actuadores cuasi directos (reducción baja) y los elásticos en serie (SEA) permiten medir y limitar la fuerza de contacto.

Par pico y par eficaz

El pico debe caber en el par máximo del motor; el calentamiento depende del par eficaz $T_{RMS} = \sqrt{(\Sigma T_k^2 \cdot t_k / \Sigma t_k)}$, que debe quedar bajo el par continuo.



Salida de dimensionamiento_eje.py.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Par del PMSM	$T = 1{,}5 \cdot p \cdot \lambda_{PM} \cdot i_q$	p pares de polos.
Reductor armónico	$i = Z_{flex} / (Z_{circ} - Z_{flex})$	200/(202 − 200) = 100.
Par en el motor	$T_m = J_m \cdot \alpha \cdot i + (J_L \cdot \alpha + T_{grav})/(i \cdot \eta)$	α: aceleración de la articulación.
Par eficaz	$T_{RMS} = \sqrt{(\Sigma T_k^2 \cdot t_k / \Sigma t_k)}$	Incluye las pausas.
Relación óptima	$i_{ópt} = \sqrt{(J_L / J_m)}$	Igual inercia a ambos lados.

Ejemplo resuelto: el motor del hombro

1. Pieza de 10 kg a 0,6 m y brazo de 12 kg (barra de 0,6 m): $T_{grav} = 9{,}81 \cdot (6 + 3{,}6) = 94{,}2$ N·m.
2. Inercia: 10·0,36 + 12·0,36/3 = 5,04 kg·m²; reductor 100:1 con η = 0,8; $J_m = 1{,}5 \cdot 10^{-4}$ kg·m².
3. Acelerando a 3 rad/s²: $T_m = 0{,}045 + (15{,}12 + 94{,}2)/80 = 1{,}41$ N·m.
4. Ciclo de 2,4 s (acelera, crucero, frena, sostiene): T_{RMS} = 1,18 N·m.

Pico 1,41 N·m contra 3,8 N·m disponibles; par eficaz 1,18 N·m contra 1,3 N·m continuos (9 % de margen). Lo limita la gravedad, no la aceleración: un contrapeso o un resorte compensador ayudaría (dimensionamiento_eje.py).

Actuadores para robótica

	Eléctrico	Neumático	Hidráulico
Densidad de potencia	Media	Baja	Muy alta
Control de posición	Excelente	Difícil (aire compresible)	Bueno
Ancho de banda	Alto	Bajo	Alto
Mantención	Baja	Media (aire seco)	Alta (fugas, filtros)

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

dimensionamiento_eje.py — reductor armónico, par gravitacional, inercia reflejada, par pico y par eficaz de un eje.

```
"""
dimensionamiento_eje.py — Capítulo 06: par pico, par eficaz (RMS) y relación de reducción
del eje de hombro de un robot con reductor armónico.
"""
import math

G = 9.81
# Carga: pieza de 10 kg a 0,6 m y brazo de 12 kg (barra uniforme de 0,6 m)
M_PIEZA, R_PIEZA = 10.0, 0.60
M_BRAZO, L_BRAZO = 12.0, 0.60
J_MOTOR = 1.5e-4          # kg·m²
I_RED, ETA = 100, 0.80    # reductor armónico 100:1
ALFA = 3.0                # rad/s² en la articulación
T_CONT, T_PICO = 1.3, 3.8 # N·m del servomotor (catálogo)


def armonico(z_flex, z_circular):
    """Relación con la corona circular fija: i = Z_flex / (Z_circ − Z_flex)."""
    return z_flex / (z_circular - z_flex)


def par_motor(alfa_carga, t_grav, j_carga):
    alfa_motor = alfa_carga * I_RED
    return J_MOTOR * alfa_motor + (j_carga * alfa_carga + t_grav) / (I_RED * ETA)


if __name__ == '__main__':
    print(f'Reductor armónico 200/202 dientes: i = {armonico(200, 202):.0f}:1')

    t_grav = G * (M_PIEZA * R_PIEZA + M_BRAZO * L_BRAZO / 2)
    j_carga = M_PIEZA * R_PIEZA ** 2 + M_BRAZO * L_BRAZO ** 2 / 3
    print(f'Par gravitacional (brazo horizontal): {t_grav:.1f} N·m')
    print(f'Inercia de la carga: {j_carga:.2f} kg·m²  -> reflejada {j_carga / I_RED ** 2 * 1e4:.2f}e-4 kg·m²')
    print(f'Relación óptima sqrt(J_carga/J_motor) = {math.sqrt(j_carga / J_MOTOR):.0f}:1')

    tramos = [  # nombre, duración s, aceleración de la carga rad/s²
        ('acelera', 0.4, +ALFA), ('velocidad cte', 0.6, 0.0),
        ('frena', 0.4, -ALFA), ('sostiene', 1.0, 0.0)]
    suma, total = 0.0, 0.0
    print('\nTramo          t (s)   par motor (N·m)')
    for nombre, t, a in tramos:
        tm = par_motor(a, t_grav, j_carga)
        suma += tm ** 2 * t
        total += t
        print(f'  {nombre:<13s} {t:4.1f}    {tm:6.3f}')
    t_rms = math.sqrt(suma / total)
```

```
    t_max = par_motor(ALFA, t_grav, j_carga)
    print(f'\nPar pico {t_max:.2f} N·m (límite {T_PICO}) -> {"OK" if t_max <= T_PICO else "EXCEDE"}')
    print(f'Par eficaz {t_rms:.2f} N·m (continuo {T_CONT}) -> margen {100 * (1 - t_rms / T_CONT):.0f} %')
```

Errores frecuentes

- Dimensionar solo por el par pico y quemar el motor por calentamiento.
- Olvidar dividir la inercia de la carga por i^2.
- No considerar el rendimiento del reductor.

Autoevaluación

1. ¿Qué corriente produce par en un PMSM con FOC?
2. ¿Qué relación da un armónico de 160 y 162 dientes?
3. ¿Qué limita el calentamiento del motor?

Respuestas. 1. i_q. **2.** 80:1. **3.** El par eficaz (RMS) del ciclo.

Resumen y cierre

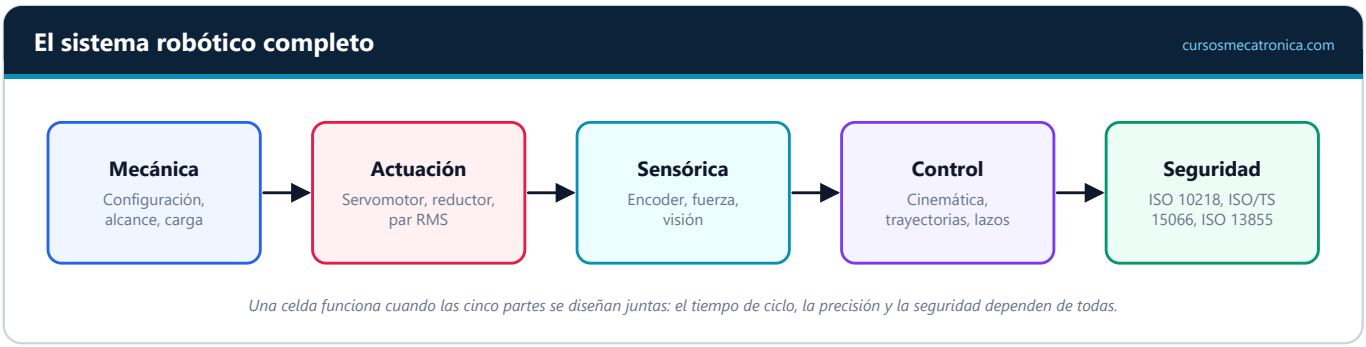
Cierre · En el video: **minuto 1:31:18** · Duración 0:50

Objetivo. Repasar cómo se conectan las partes de un sistema robótico.

Ideas clave

Lo que se recorrió

La configuración fija el espacio de trabajo; los sensores y el controlador cierran los lazos y vigilan la seguridad; la cinemática traduce entre articulaciones y herramienta; las trayectorias definen cómo se mueve y los actuadores lo hacen posible.



Síntesis del minicurso.

Para seguir

Repite los programas con los datos de un robot real (catálogo de un brazo de 6 ejes) y compara alcance, carga en la muñeca y par eficaz con lo que indica el fabricante.

Fórmulas del minicurso

Tema	Relación
Movilidad	$M = 6(n - 1 - j) + \Sigma f_i$
Takt	tiempo disponible / demanda
Separación	$S = 1600 \cdot T + C$
Inversa 2R	$c_2 = (x^2 + y^2 - l_1^2 - l_2^2)/(2l_1l_2)$
Repetibilidad	$RP = \bar{l} + 3S_l$
Par eficaz	$T_{RMS} = \sqrt{(\Sigma T^2 t/\Sigma t)}$

Autoevaluación

1. ¿Qué dos verificaciones requiere un servomotor?
2. ¿Qué norma fija la distancia de un escáner de seguridad?

Respuestas. 1. Par pico y par eficaz. **2.** ISO 13855.