MANUAL COMPLETO DEL MINICURSO · CURSOSMECATRONICA.COM

Mecatrónica Automotriz

Arquitectura eléctrica y electrónica, unidades de control y redes, sensores y actuadores, transmisión, frenos ABS, control de estabilidad, inyección diésel, dirección, climatización, compatibilidad electromagnética y diagnóstico a bordo.

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. Los programas usan solo la biblioteca estándar de Python 3.8 o superior y no necesitan interfaz CAN ni vehículo: modelan cada sistema y reproducen las cifras de los ejemplos resueltos.

Contenido

N.º	Capítulo	Video
00	**Introducción al curso** Recorrido del curso y cómo usar los manuales y programas.	min 0:00
01	**Sistemas mecatrónicos y arquitectura eléctrica y electrónica zonal** Lazo mecatrónico, modelo en V, dominios del vehículo, coordinador de movimiento y arquitectura zonal.	min 0:52
02	**Unidades de control electrónico (ECU) y AUTOSAR** Entradas, conversión A/D, etapas de potencia, microcontrolador, memorias, punto fijo, tareas, watchdog y AUTOSAR.	min 15:47
03	**Redes automotrices: topologías, modelo OSI y arquitectura multibus** Multiplexado, topologías, acceso al medio, TDMA, OSI, clases SAE, pasarelas y Ethernet con TSN.	min 30:18
04	**Buses CAN, LIN, FlexRay, MOST y Ethernet automotriz** Capa física CAN, arbitraje, trama y CRC, CAN FD y XL, LIN, FlexRay y Ethernet automotriz.	min 44:56
05	**Sensores automotrices: principios físicos y protocolos SENT y PSI5** Integración de sensores, SENT y PSI5, errores de medición y principios inductivos, Hall, AMR, capacitivos y piezoeléctricos.	min 1:00:13
06	**Sensores de posición, velocidad, presión, caudal y oxígeno** Velocidad de rueda, pedal y ángulo de dirección, MEMS, presión, caudalímetro HFM, detonación y sondas lambda.	min 1:15:42
07	**Actuadores eléctricos y electrohidráulicos: PWM y motores BLDC** Conmutación PWM, puente H, motores DC y BLDC, electroválvulas y reguladores proporcionales con control de corriente.	min 1:30:49
08	**Control electrónico de la transmisión: AT, DCT, CVT, DHT y TCU** Convertidor de par, trenes planetarios, cambio con solape, DCT, CVT, DHT y TCU integrada.	min 1:45:59
09	**Frenos ABS, deslizamiento, frenado regenerativo y control de tracción** Curva μ–deslizamiento, ciclos de regulación ABS, GMA, frenado regenerativo y TCS.	min 2:00:38
10	**Control de estabilidad, ADAS y frenos by-wire** Modelo de vía única, guiñada, ESP, EBD, frenado de emergencia y sistemas brake-by-wire.	min 2:14:55
11	**Inyección diésel common rail: bombas e inyectores** Riel acumulador, presión y caudal, inyección múltiple, bombas CP e inyectores de solenoide y piezoeléctricos.	min 2:29:19
12	**Control electrónico diésel y postratamiento de emisiones** Gestión guiada por par, VGT, EGR, límite de humos, DOC, DPF y SCR bajo Euro 7.	min 2:43:51
13	**Dirección eléctrica y by-wire, climatización y seguridad** Dirección activa, EPS y steer-by-wire, antipinzamiento, HVAC con bomba de calor y acceso UWB.	min 2:58:14
14	**Compatibilidad electromagnética, plataformas de 800 V y diagnóstico a bordo** Acoplamientos CEM, transitorios, ensayos, DTC, OBD, UDS y DoIP.	min 3:12:39

15	**Resumen y cierre** Síntesis del curso.	min 3:27:08

Curso gratuito en video: **cursosmecatronica.com/cursos/mecatronica-automotriz/**

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Introducción al curso

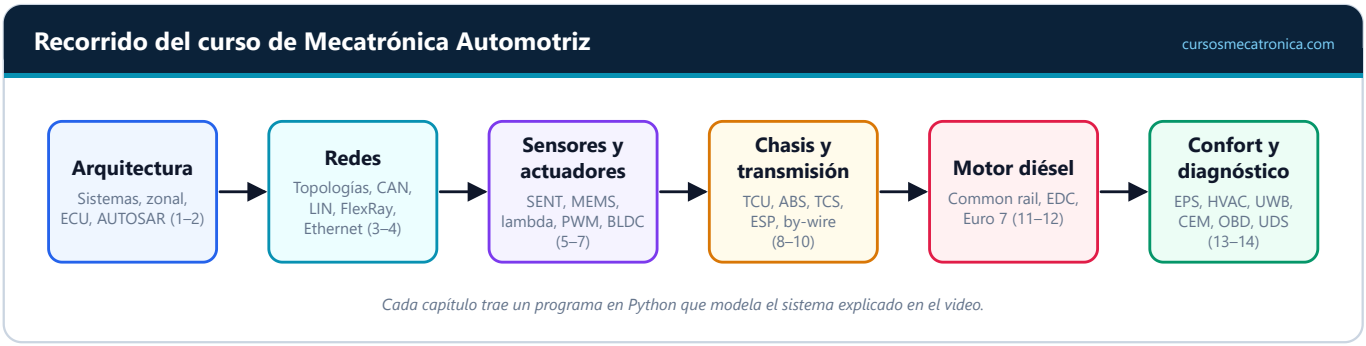
Introducción · En el video: minuto 0:00 · Duración 0:52

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

Ideas clave

El vehículo como sistema mecatrónico

Un automóvil actual tiene decenas de unidades de control conectadas por varias redes, cientos de sensores y actuadores, y software que coordina motor, transmisión, frenos y dirección. El curso recorre esa cadena: arquitectura, unidades de control y redes; sensores y actuadores; y sus aplicaciones en transmisión, frenos, estabilidad, inyección diésel, dirección, confort y diagnóstico.



Estructura del curso.

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`. Los programas usan solo la biblioteca estándar: simulan buses, sensores, lazos de control y diagnósticos sin necesidad de un vehículo ni de una interfaz CAN.

Autoevaluación

1. ¿Hace falta hardware para los programas del curso?

Respuestas. 1. No: son modelos y algoritmos que corren en cualquier PC con Python 3.8 o superior.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Sistemas mecatrónicos y arquitectura eléctrica y electrónica zonal

Capítulo 01 · En el video: **minuto 0:52** · Duración 14:55

Objetivo. Describir cómo se organiza la electrónica de un vehículo y cómo se arbitran pedidos de varias funciones.

Ideas clave

El lazo mecatrónico

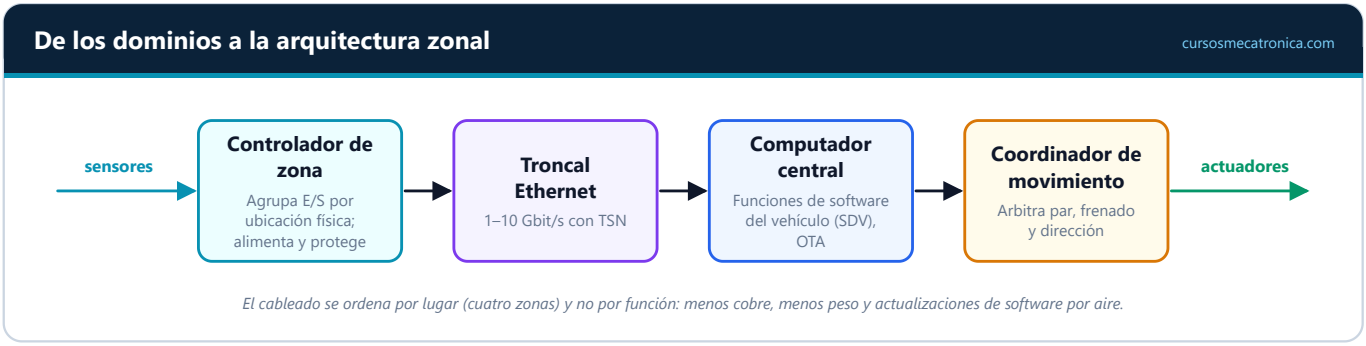
Sensores, unidad de control, actuadores y el proceso mecánico forman un lazo cerrado. El diseño se hace en conjunto: modelos en espacio de estados acoplan las partes mecánica, eléctrica y de software desde el principio, y la validación virtual adelanta errores que antes aparecían en los prototipos.

Modelo en V y seguridad funcional

El lado izquierdo de la V baja de requisitos del vehículo a sistemas, componentes y código; el derecho sube con pruebas de componente, integración y vehículo. ISO 26262 asigna a cada función un nivel ASIL (A a D) según la gravedad, la exposición y la controlabilidad de su falla.

Más electrónica en cada generación

La proporción del valor del vehículo que corresponde a electrónica y software ha crecido de forma sostenida desde los años setenta, y en los eléctricos es aún mayor. Eso llevó primero a agrupar las ECU por dominios (tren motriz, chasis, carrocería, infoentretenimiento, asistencia).



Arquitectura eléctrica y electrónica zonal.

Coordinador de movimiento

El dominio de movimiento tiene una jerarquía: el conductor y los asistentes piden, un coordinador calcula el vector global de fuerzas (longitudinal, lateral y de guiñada) y lo reparte entre motor, frenos y dirección. Así nacieron arquitecturas funcionales como CARTRONIC.



Cadena de arbitraje del ejemplo del capítulo 01.

Vehículo definido por software

Con computadores centrales y zonas, las funciones se actualizan por aire (OTA). Los vehículos eléctricos agregan plataformas de 800 V para cargar más rápido con menos corriente.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Arbitraje de deseos	$T = \text{máx}(T_{\text{conductor}}, T_{\text{ACC}})$	El conductor puede anular al ACC pisando más.
Arbitraje de límites	$T = \text{mín}(T, T_{\text{lím},1}, T_{\text{lím},2}, \ldots)$	Protecciones y TCS.
Pisos	$T = \text{máx}(T, T_{\text{mín}})$	Por ejemplo, regulación del freno motor (MSR).
Riesgo ISO 26262	$\text{ASIL} = f(S, E, C)$	Severidad, exposición, controlabilidad.

Ejemplo resuelto: arbitraje de par

1. El conductor pide 180 N·m; el ACC pide 120 N·m: gana el máximo, 180.
2. La transmisión limita a 150 N·m durante un cambio.
3. El control de tracción detecta patinamiento y limita a 90 N·m.
4. El ESP fija un piso de 40 N·m para que el freno motor no bloquee las ruedas: no actúa porque 90 > 40.

Se entregan 90 N·m. arbitraje_y_zonal.py también compara el cableado de 24 nodos: 47,7 m con una ECU central frente a 29,9 m con cuatro zonas (37 % menos).

Dominios funcionales del vehículo

Dominio	Funciones	Exigencia
Tren motriz	Motor, transmisión, batería e inversor	Tiempo real, temperatura
Chasis	Frenos, dirección, suspensión	Seguridad (ASIL C/D)
Carrocería y confort	Luces, puertas, climatización	Muchos nodos, bajo costo
Infoentretenimiento	Pantallas, audio, conectividad	Ancho de banda
Asistencia (ADAS)	Cámaras, radar, fusión	Cómputo y ancho de banda

Código del capítulo

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

arbitraje_y_zonal.py — arbitraje de par con deseos, límites y pisos; largo de cableado centralizado frente a zonal.

```
"""
arbitraje_y_zonal.py — Capítulo 01: coordinador de movimiento (arbitraje de par) y largo de
cableado de una arquitectura centralizada frente a una zonal.
"""
import math


def arbitrar(deseo_conductor, pedidos):
    """pedidos: (fuente, tipo, valor). 'deseo' compite por máximo (anulación del conductor);
    'max' limita hacia abajo; 'min' asegura un piso (p. ej. arrastre del motor en el ESP)."""
    deseo = deseo_conductor
    traza = [('conductor', deseo)]
    for fuente, tipo, valor in pedidos:
        if tipo == 'deseo':
            deseo = max(deseo, valor)
        elif tipo == 'max':
            deseo = min(deseo, valor)
        elif tipo == 'min':
            deseo = max(deseo, valor)
        traza.append((fuente, deseo))
    return deseo, traza


# Posiciones (x, y) en metros de 24 sensores y actuadores del vehículo
REJILLA = [(x, y) for x in (0.3, 1.2, 2.4, 3.3, 4.2) for y in (-0.7, 0.0, 0.7)]
EXTRA = [(0.2, -0.5), (0.2, 0.5), (4.4, -0.5), (4.4, 0.5), (1.8, -0.8), (1.8, 0.8),
         (2.8, -0.8), (2.8, 0.8), (3.9, 0.0)]
NODOS = REJILLA + EXTRA


def largo(a, b):
    return abs(a[0] - b[0]) + abs(a[1] - b[1])      # cable sigue la carrocería


def cableado_central(ecu=(1.4, 0.0)):
    return sum(largo(n, ecu) for n in NODOS)


def cableado_zonal(zonas, central=(1.4, 0.0)):
    total = sum(min(largo(n, z) for z in zonas) for n in NODOS)
    troncal = sum(largo(z, central) for z in zonas)
    return total, troncal


if __name__ == '__main__':
    final, traza = arbitrar(180, [('ACC', 'deseo', 120), ('transmisión', 'max', 150),
                                  ('TCS', 'max', 90), ('ESP-MSR', 'min', 40)])
    print('Arbitraje de par (N·m):')
    for fuente, valor in traza:
        print(f'  después de {fuente:<12s}: {valor:5.0f}')
    print(f'  Par entregado al motor: {final:.0f} N·m')

    c = cableado_central()
    zonas = [(0.6, -0.6), (0.6, 0.6), (3.6, -0.6), (3.6, 0.6)]
    z, troncal = cableado_zonal(zonas)
    print(f'\n{len(NODOS)} nodos: arquitectura centralizada {c:.1f} m de cable')
    print(f'Zonal (4 zonas): {z:.1f} m locales + {troncal:.1f} m de troncal Ethernet'
          f' = {z + troncal:.1f} m  ({100 * (1 - (z + troncal) / c):.0f} % menos)')
```

Errores frecuentes

- Arbitrar todo por prioridad fija en vez de por tipo de pedido (deseo, límite, piso).
- Pensar la arquitectura zonal como un cambio solo de software: también cambia el mazo de cables y la alimentación.
- Olvidar que cada función tiene un ASIL que condiciona su hardware y su software.

Autoevaluación

1. ¿Qué gana la arquitectura zonal?
2. Si el conductor pide 180 N·m y el TCS limita a 90, ¿cuánto se entrega?
3. ¿Qué norma define los niveles ASIL?

Respuestas. 1. Menos cable y peso, y funciones centralizadas actualizables por aire. **2.** 90 N·m. **3.** ISO 26262.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Unidades de control electrónico (ECU) y AUTOSAR

Capítulo 02 · En el video: **minuto 15:47** · Duración 14:31

Objetivo. Seguir una señal desde el sensor hasta el actuador dentro de una ECU y verificar su software en tiempo real.

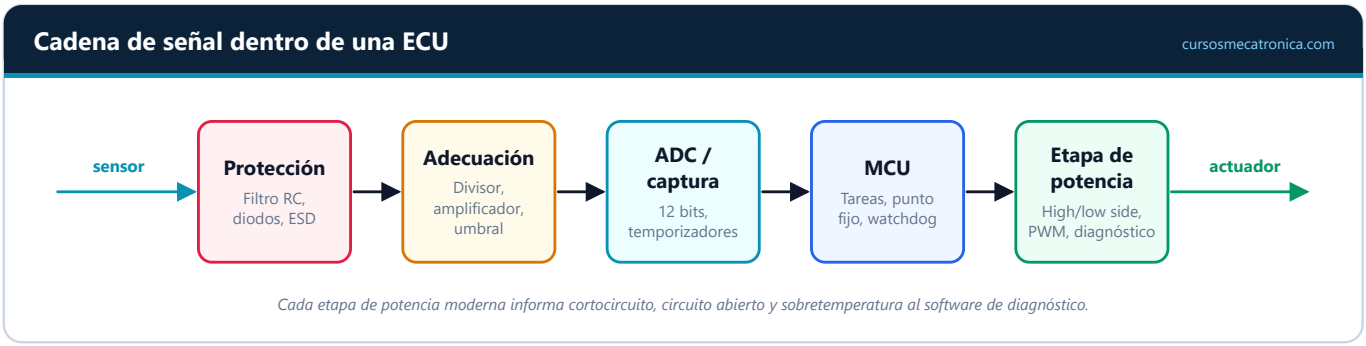
Ideas clave

Diseño constructivo

Una ECU soporta de −40 a 125 °C (más cerca del motor), vibraciones, humedad, sal y perturbaciones eléctricas. Por eso usa carcasas selladas, componentes calificados AEC-Q100/Q200 y protecciones en cada pin.

Entradas y conversión

Las entradas analógicas pasan por filtros y divisores hacia un ADC de 10 a 12 bits; las de frecuencia (régimen, velocidad) van a temporizadores de captura. Un sensor NTC con resistencia de pull-up es el caso típico.



Bloques de hardware de una unidad de control.

Etapas de potencia

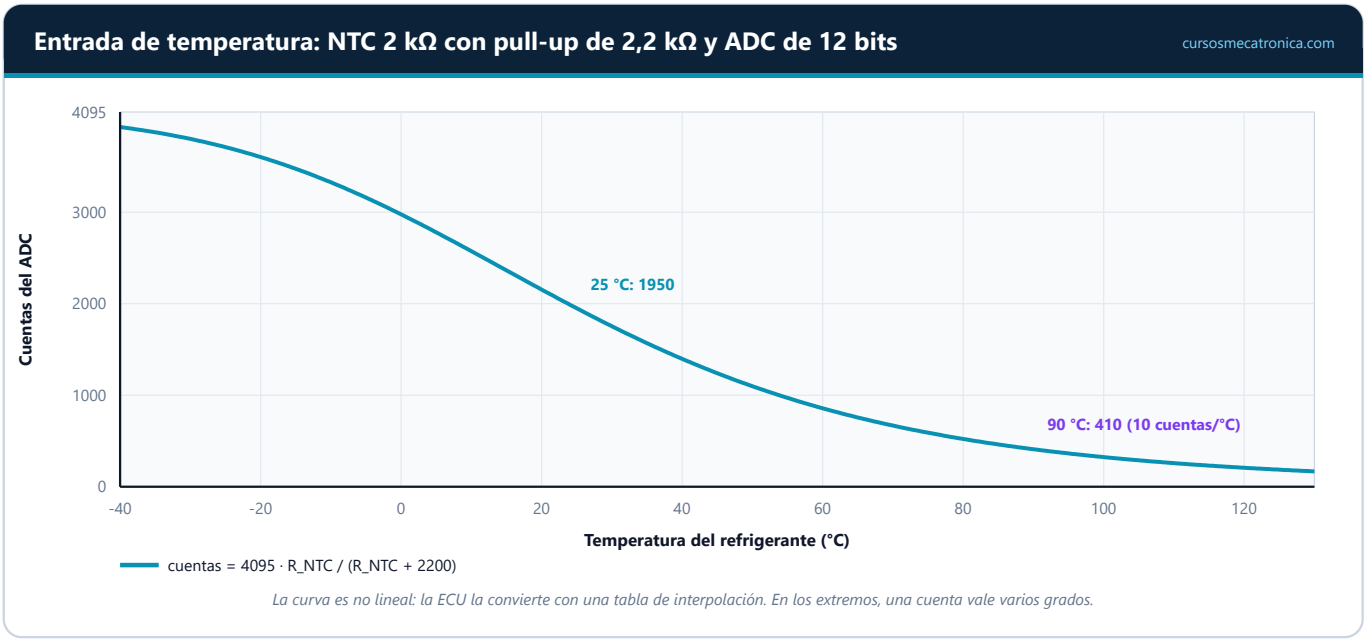
Interruptores de lado alto (al positivo) o bajo (a masa) con MOSFET inteligentes manejan lámparas, relés y válvulas; el PWM regula la potencia media. Cada etapa diagnostica cortocircuitos, circuitos abiertos y sobretemperatura; la forma de la corriente (firma) permite detectar si una válvula se movió.

Microcontrolador y memorias

Núcleos múltiples en paralelo (lockstep) para seguridad, flash para el programa y los mapas de calibración, RAM y EEPROM emulada para adaptaciones. El sincronismo angular con el cigüeñal dispara inyección y encendido en el grado exacto.

Punto fijo y tareas

Muchos cálculos usan enteros escalados (Q15, Q31) por velocidad y determinismo; hay que cuidar desbordes y saturar. El sistema operativo ejecuta tareas periódicas (1, 10, 100 ms) por prioridad; un watchdog de ventana reinicia la ECU si el software se cuelga o se acelera.



Salida de ecu_entradas_y_tareas.py.

AUTOSAR y validación

AUTOSAR Classic separa aplicación, entorno de ejecución (RTE) y software de base, para reutilizar componentes entre fabricantes; AUTOSAR Adaptive se usa en computadores centrales con POSIX y Ethernet. El software se calibra con ASAM MCD (XCP, A2L) y se prueba en bancos HIL que simulan el vehículo alrededor de la ECU real.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Divisor NTC	$V = V_{ref} \cdot R_{NTC} / (R_{NTC} + R_{pull})$	NTC a masa.
Modelo β	$R(T) = R_{25} \cdot e^{\beta(1/T - 1/298{,}15)}$	T en kelvin.
ADC	$n = V / V_{ref} \cdot (2^N - 1)$	12 bits: 0 a 4095.
Producto Q15	$(a \cdot b) >> 15$	Con saturación a ±32 767.
Planificabilidad (RMS)	$U = \Sigma C_i / T_i \leq n(2^{1/n} - 1)$	Cota suficiente de Liu y Layland.

Ejemplo resuelto: resolución de la temperatura del motor

1. NTC de 2 kΩ a 25 °C (β = 3500) con pull-up de 2,2 kΩ a 5 V y ADC de 12 bits.
2. A 25 °C: 1950 cuentas; a 90 °C, R = 245 Ω y 410 cuentas.
3. Entre 89,5 y 90,5 °C cambian 10 cuentas: 0,1 °C por cuenta.
4. Tres tareas (0,2/1 ms, 2,5/10 ms, 20/100 ms) cargan la CPU al 65 %, bajo la cota de 78 %.

La medición es fina donde importa (zona de trabajo) y la carga de CPU es planificable. ecu_entradas_y_tareas.py simula además un watchdog de ventana que reinicia la ECU cuando una tarea se atrasa a 14,5 ms.

AUTOSAR Classic frente a Adaptive

	Classic	Adaptive
Hardware	Microcontrolador	Procesador de alto rendimiento
Sistema operativo	OSEK/AUTOSAR OS, estático	POSIX, dinámico
Comunicación	CAN, LIN, FlexRay	Ethernet orientado a servicios (SOME/IP)
Uso	Motor, frenos, carrocería	Computador central, ADAS, OTA

Código del capítulo

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

ecu_entradas_y_tareas.py — NTC con ADC de 12 bits, producto Q15 con saturación, carga de CPU y watchdog de ventana.

```
"""
ecu_entradas_y_tareas.py — Capítulo 02: entrada NTC con ADC de 12 bits, aritmética Q15,
carga de CPU con planificación de prioridades fijas y watchdog de ventana.
"""
import math

VREF, R_PULL, R25, BETA = 5.0, 2200.0, 2000.0, 3500.0


def r_ntc(temp_c):
    return R25 * math.exp(BETA * (1 / (temp_c + 273.15) - 1 / 298.15))


def cuentas_adc(temp_c, bits=12):
    v = VREF * r_ntc(temp_c) / (r_ntc(temp_c) + R_PULL)    # NTC abajo, pull-up arriba
    return round(v / VREF * (2 ** bits - 1))


def q15(x):
    return max(-32768, min(32767, round(x * 32768)))


def q15_mul(a, b):
    return max(-32768, min(32767, (a * b) >> 15))


def carga_cpu(tareas):
    u = sum(c / t for c, t in tareas)
    n = len(tareas)
    return u, n * (2 ** (1 / n) - 1)


def watchdog(disparos_ms, ventana=(8.0, 12.0)):
    """Watchdog de ventana: cada disparo debe llegar entre 8 y 12 ms después del anterior."""
    for previo, actual in zip(disparos_ms, disparos_ms[1:]):
        dt = actual - previo
        if not ventana[0] <= dt <= ventana[1]:
            return f'RESET en t = {actual:.1f} ms (intervalo {dt:.1f} ms)'
    return 'sin reset'


if __name__ == '__main__':
    print('NTC 2 kΩ a 25 °C, B = 3500, pull-up 2,2 kΩ, ADC 12 bits')
    for t in (-40, 25, 90, 120):
```

```
        print(f'  {t:4d} °C -> R = {r_ntc(t):8.0f} Ω  cuentas = {cuentas_adc(t)}')
    sens = cuentas_adc(89.5) - cuentas_adc(90.5)
    print(f'  Resolución a 90 °C: {sens} cuentas por °C -> {1 / sens:.3f} °C por cuenta')

    a, b = q15(0.75), q15(-0.4)
    print(f'\nQ15: 0,75 × (−0,4) = {q15_mul(a, b) / 32768:.6f} (exacto −0,3)')
    print(f'Q15 saturado: 0,9 + 0,9 -> {min(32767, q15(0.9) + q15(0.9)) / 32768:.5f}')

    tareas = [(0.2, 1.0), (2.5, 10.0), (20.0, 100.0)]     # (ms de cómputo, periodo ms)
    u, cota = carga_cpu(tareas)
    print(f'\nCarga de CPU U = {100 * u:.1f} %; cota de Liu-Layland para 3 tareas = {100 * cota:.1f} %'
          f' -> {"planificable" if u <= cota else "revisar"}')
    print('Watchdog, disparos cada 10 ms:', watchdog([0, 10, 20, 30, 40]))
    print('Watchdog, una tarea se cuelga  :', watchdog([0, 10, 20, 34.5, 44.5]))
```

Errores frecuentes

- Multiplicar en Q15 sin saturar y obtener un cambio de signo por desborde.
- Dimensionar el watchdog solo con tiempo máximo: un software que corre "demasiado rápido" también es una falla.
- Suponer que la curva NTC es lineal.

Autoevaluación

1. ¿Qué hace un watchdog de ventana si se le dispara antes de tiempo?
2. ¿Qué capa de AUTOSAR comunica la aplicación con el software de base?
3. ¿Qué prueba un banco HIL?

Respuestas. 1. Reinicia la ECU, igual que si llega tarde. **2.** El RTE. **3.** La ECU real con el vehículo simulado a su alrededor.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Redes automotrices: topologías, modelo OSI y arquitectura multibus

Capítulo 03 · En el video: **minuto 30:18** · Duración 14:38

Objetivo. Elegir topología y bus para cada dominio y calcular la carga de una red CAN.

Ideas clave

Del cable dedicado al bus

Cablear cada señal punto a punto multiplicaba conectores y kilos de cobre. Un bus serie multiplexado transmite muchas señales por un par trenzado y permite compartir sensores entre ECU.

Topologías

Bus lineal (CAN, LIN): simple, pero un corto lo afecta entero. Estrella (Ethernet, FlexRay activo): aísla fallas en un acoplador. Anillo (MOST): cada nodo repite. Árbol: combinación jerárquica.

Acceso al medio

Maestro-esclavo (LIN): el maestro pregunta y cada esclavo responde. CSMA con arbitraje no destructivo (CAN): habla quien quiere y gana la prioridad más alta. TDMA (FlexRay, TSN): cada nodo tiene su ranura de tiempo, lo que hace el sistema determinista y componible.

Modelo OSI

Los buses automotrices usan sobre todo las capas física, de enlace y de aplicación; el transporte (ISO-TP) aparece cuando un mensaje no cabe en una trama. SAE clasifica las redes por velocidad: clase A (< 10 kbit/s, LIN), B (10–125 kbit/s), C (125 kbit/s–1 Mbit/s, CAN de alta velocidad) y D (> 1 Mbit/s, FlexRay, Ethernet).



Partición de redes por dominio.

Pasarelas

La pasarela enruta mensajes entre redes, traduce formatos, filtra como cortafuegos y gestiona el reposo y el despertar de cada red para no descargar la batería con el auto estacionado.

Ethernet con TSN

Las cámaras y los computadores centrales necesitan gigabits. Time-Sensitive Networking agrega a Ethernet sincronización de relojes y reservas de tiempo para tráfico de control.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Bits de una trama CAN (peor caso)	$b = 47 + 8 \cdot DLC + \lfloor (34 + 8 \cdot DLC - 1)/4 \rfloor$	135 bits con 8 bytes.
Carga del bus	$U = \Sigma(b_i/T_i)$ / bitrate	Conviene < 50–60 %.
Ranura TDMA	$t_{ranura} = ciclo$ / $n_{ranuras}$	Latencia acotada.

Ejemplo resuelto: carga del CAN de tren motriz

1. 14 mensajes con periodos de 10 a 1000 ms (la mayoría de 8 bytes).
2. Cada trama de 8 bytes ocupa hasta 135 bits.
3. Suma de bits por segundo dividida por 500 kbit/s: $U = 14{,}0$ %.
4. A 250 kbit/s la carga sube a 27,9 %.

Hay margen para más mensajes y para los reintentos tras errores. carga_bus_y_pasarela.py simula también una pasarela que reenvía, descarta o bloquea tramas según su origen.

Buses automotrices

Bus	Velocidad	Acceso	Uso
LIN	≤ 20 kbit/s	Maestro-esclavo	Espejos, asientos, sensores de lluvia
CAN / CAN FD	0,5–1 / 2–8 Mbit/s	CSMA con arbitraje	Tren motriz, chasis, carrocería
FlexRay	10 Mbit/s	TDMA + dinámico	Chasis activo, by-wire
MOST	25–150 Mbit/s	Anillo síncrono	Audio (en retirada)
Ethernet automotriz	100 Mbit/s–10 Gbit/s	Conmutado, TSN	Troncal, cámaras, ADAS

Código del capítulo

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

carga_bus_y_pasarela.py — carga de un bus CAN con 14 mensajes y pasarela con rutas y cortafuegos.

```
"""
carga_bus_y_pasarela.py — Capítulo 03: carga de un bus CAN y pasarela con tabla de
enrutamiento y cortafuegos entre dominios.
"""


def bits_trama(dlc):
    """Trama CAN estándar (11 bits) en el peor caso de relleno de bits."""
    return 47 + 8 * dlc + (34 + 8 * dlc - 1) // 4


def carga_bus(mensajes, bitrate):
    bits_s = sum(bits_trama(dlc) * 1000 / periodo_ms for _, dlc, periodo_ms in mensajes)
    return bits_s / bitrate


TREN_MOTRIZ = [  # (ID, DLC, periodo ms)
```

```
    (0x0A0, 8, 10), (0x0A8, 8, 10), (0x0C0, 8, 10), (0x110, 8, 20), (0x120, 8, 20),
    (0x1A0, 6, 20), (0x1F0, 8, 50), (0x200, 8, 50), (0x280, 4, 100), (0x300, 8, 100),
    (0x3C0, 8, 100), (0x400, 8, 200), (0x480, 2, 500), (0x500, 8, 1000)]

RUTAS = {0x0A0: 'carroceria', 0x1A0: 'chasis', 0x300: 'carroceria'}   # ID -> red destino
DIAGNOSTICO = range(0x7E0, 0x7F0)


def pasarela(id_trama, origen, sesion_diagnostico=False):
    if id_trama in DIAGNOSTICO and not sesion_diagnostico:
        return 'bloqueada (diagnóstico sin sesión autenticada)'
    if origen == 'obd':
        if id_trama in DIAGNOSTICO:
            return 'a tren motriz'
        return 'bloqueada (desde el conector solo se admite diagnóstico)'
    destino = RUTAS.get(id_trama)
    return f'reenviada a {destino}' if destino else 'descartada (sin ruta)'


if __name__ == '__main__':
    print(f'Bits por trama de 8 bytes (peor caso): {bits_trama(8)}')
    for rate in (500_000, 250_000):
        print(f'Carga del CAN de tren motriz a {rate // 1000} kbit/s:'
              f' {100 * carga_bus(TREN_MOTRIZ, rate):.1f} %')
    print('\nPasarela:')
    pruebas = [(0x0A0, 'tren', False), (0x1A0, 'tren', False), (0x123, 'tren', False),
               (0x0A0, 'obd', False), (0x7E0, 'obd', False), (0x7E0, 'obd', True)]
    for t, o, s in pruebas:
        print(f'  0x{t:03X} desde {o:<5s}: {pasarela(t, o, s)}')
```

Errores frecuentes

- Calcular la carga sin los bits de relleno.
- Dejar que el conector OBD tenga acceso directo a las redes internas.
- Usar CAN para video: el ancho de banda no alcanza.

Autoevaluación

1. ¿Qué método de acceso usa LIN?
2. ¿Qué hace determinista a FlexRay?
3. ¿Qué carga tiene el bus del ejemplo a 500 kbit/s?

Respuestas. 1. Maestro-esclavo. **2.** Las ranuras de tiempo fijas (TDMA). **3.** 14,0 %.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Buses CAN, LIN, FlexRay, MOST y Ethernet automotriz

Capítulo 04 · En el video: **minuto 44:56** · Duración 15:17

Objetivo. Explicar cómo CAN resuelve colisiones sin perder tiempo y cómo se detectan los errores de transmisión.

Ideas clave

Capa física CAN

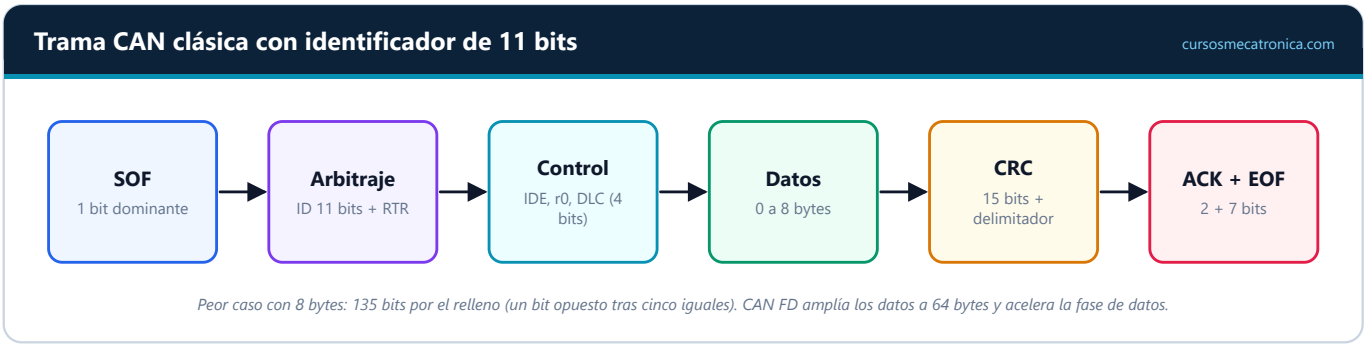
Par trenzado CAN_H/CAN_L con terminaciones de 120 Ω en los extremos. En recesivo ambos quedan en ~2,5 V; en dominante CAN_H sube a ~3,5 V y CAN_L baja a ~1,5 V. La señal diferencial rechaza el ruido común.

Arbitraje bit a bit

Los identificadores no son direcciones sino prioridades del contenido. Todos transmiten a la vez; el bit dominante (0) se impone en el cable. Quien envía un 1 y lee un 0 deja de transmitir: el mensaje de ID menor sigue sin perder ni un bit.

Trama y errores

Tras cinco bits iguales se inserta uno opuesto (relleno) para mantener la sincronización. El CRC-15 detecta errores; el receptor confirma en el bit ACK. Los contadores de errores llevan a un nodo defectuoso a "bus off" para que no bloquee la red.



Campos de la trama del capítulo 04.

CAN FD y CAN XL

CAN FD conserva el arbitraje a velocidad normal y acelera la fase de datos (2 a 8 Mbit/s) con hasta 64 bytes por trama. CAN XL llega a 2048 bytes y unos 20 Mbit/s, como puente hacia Ethernet.

LIN, FlexRay y MOST

LIN es un bus de un cable de bajo costo con un maestro que agenda los mensajes. FlexRay combina un segmento estático (TDMA) y uno dinámico, con relojes sincronizados y guardián de bus. MOST, un anillo óptico de audio, fue reemplazado por Ethernet.

Ethernet automotriz

100BASE-T1 y 1000BASE-T1 usan un solo par trenzado; los conmutadores con TSN garantizan latencia. Con ISO-TP, los mensajes largos de diagnóstico se segmentan en una primera trama y tramas consecutivas.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Polinomio CRC-15	$x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1$	0x4599.
Tiempo de bit	$t_{bit} = 1/\text{bitrate}$	2 µs a 500 kbit/s.
Largo máximo del bus	$L \approx (t_{bit}/2 - t_{nodos}) \cdot v$	El bit debe recorrer el bus ida y vuelta.
ISO-TP	$FF + \lceil (n-6)/7 \rceil \ CF$	Mensaje de n bytes en tramas de 8.

Ejemplo resuelto: arbitraje entre cuatro nodos

1. Transmiten a la vez 0x123, 0x120, 0x7FF y 0x0F0.
2. Bit 1: 0x7FF envía 1 y lee 0: se retira.
3. Bit 3: 0x123 y 0x120 envían 1 y 0x0F0 envía 0: ambos se retiran.
4. Gana 0x0F0, el identificador de mayor prioridad (menor valor).

Una trama 0x0A0 con 8 bytes lleva 13 bits de relleno y dura 124 bits = 248 µs a 500 kbit/s. 64 bytes por CAN FD (500k/2M) tardan unos 347 µs frente a 2160 µs en ocho tramas clásicas (can_arbitraje_y_crc.py).

Niveles del bus CAN de alta velocidad (ISO 11898-2)

Estado	CAN_H	CAN_L	Diferencial	Bit
Recesivo	≈ 2,5 V	≈ 2,5 V	≈ 0 V	1
Dominante	≈ 3,5 V	≈ 1,5 V	≈ 2 V	0

Código del capítulo

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

can_arbitraje_y_crc.py — arbitraje bit a bit, CRC-15, relleno de bits, duración de tramas y segmentación ISO-TP. *(extracto: primeras 60 de 83 líneas)*

```
"""
can_arbitraje_y_crc.py — Capítulo 04: arbitraje bit a bit, CRC-15, relleno de bits,
duración de tramas CAN y CAN FD y segmentación ISO-TP.
"""


def arbitraje(ids):
    """Bus de 'Y cableado': el 0 (dominante) gana. Devuelve el ganador y quién perdió en qué bit."""
    activos, perdidas = list(ids), {}
    for bit in range(10, -1, -1):
        nivel = min((i >> bit) & 1 for i in activos)
        for i in [i for i in activos if (i >> bit) & 1 != nivel]:
            perdidas[i] = 11 - bit          # posición 1 = primer bit transmitido
            activos.remove(i)
    return activos[0], perdidas


def crc15(bits):
    crc = 0
```

```
    for b in bits:
        siguiente = b ^ ((crc >> 14) & 1)
        crc = (crc << 1) & 0x7FFF
        if siguiente:
            crc ^= 0x4599
    return crc


def a_bits(valor, n):
    return [(valor >> k) & 1 for k in range(n - 1, -1, -1)]


def trama(id_std, datos):
    bits = [0] + a_bits(id_std, 11) + [0, 0, 0] + a_bits(len(datos), 4)
    for byte in datos:
        bits += a_bits(byte, 8)
    return bits + a_bits(crc15(bits), 15)


def relleno(bits):
    """Cuenta los bits de relleno: tras 5 iguales se inserta uno opuesto."""
    n, previo, racha = 0, None, 0
    for b in bits:
        if b == previo:
            racha += 1
        else:
            previo, racha = b, 1
        if racha == 5:
            n += 1
            previo, racha = 1 - b, 1
    return n


def isotp(carga):
    """Segmentación ISO 15765-2 en tramas clásicas de 8 bytes."""
    if len(carga) <= 7:
        return [[len(carga)] + carga]
    tramas = [[0x10 | (len(carga) >> 8), len(carga) & 0xFF] + carga[:6]]
    resto, sn = carga[6:], 1
    while resto:
        tramas.append([0x20 | sn] + resto[:7])
```

Errores frecuentes

- Olvidar una de las dos terminaciones de 120 Ω (reflexiones y errores).
- Asignar IDs bajos a mensajes poco importantes y retrasar los críticos.
- Medir CAN_H y CAN_L respecto de masa y no en modo diferencial.

Autoevaluación

1. ¿Qué bit gana en el bus CAN?
2. ¿Cuántos bytes lleva una trama CAN FD?
3. ¿Para qué sirve el relleno de bits?

Respuestas. 1. El dominante (0). **2.** Hasta 64. **3.** Para asegurar flancos que mantengan sincronizados a los nodos.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Sensores automotrices: principios físicos y protocolos SENT y PSI5

Capítulo 05 · En el video: **minuto 1:00:13** · Duración 15:29

Objetivo. Relacionar cada principio físico con su sensor automotriz y decodificar una trama SENT.

Ideas clave

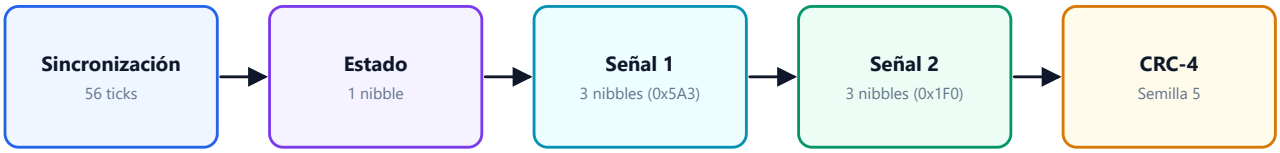
Mapa de sensores

Tren motriz (presión, temperatura, régimen, lambda), chasis (velocidad de rueda, aceleración, guiñada, ángulo de dirección), carrocería (lluvia, luz, humedad) y asistencia (radar, cámara, ultrasonido).

Niveles de integración

Del elemento sensor solo, al sensor con acondicionamiento, al sensor inteligente con ADC, autodiagnóstico y salida digital. SENT envía datos unidireccionales con pulsos de duración variable; PSI5 es un lazo de corriente de dos hilos para airbags.

Trama SENT (SAE J2716) con dos señales de 12 bits — cursosmecatronica.com



Cada nibble dura 12 + valor ticks (tick = 3 µs). La trama del ejemplo dura 193 ticks = 579 µs; un nibble alterado hace fallar el CRC.

Salida de sent_y_tolerancias.py.

Errores y fiabilidad

Offset, ganancia, deriva térmica, histéresis y cuantización se reparten en una banda de tolerancia. El peor caso suma todos; el presupuesto estadístico (RSS) los combina en cuadratura.

Ambiente y micromecánica

Temperaturas extremas, vibración, combustibles y sal. La micromecánica en silicio (MEMS) fabrica estructuras móviles junto a su electrónica.

Principios inductivos y magnéticos

Bobinas, corrientes de Foucault y resolvers miden posición sin contacto. El efecto Hall entrega una tensión proporcional al campo; la magnetorresistencia gigante (GMR) y la anisótropa (AMR) cambian su resistencia con la dirección del campo: la AMR, con estructuras "barber pole", se linealiza para medir ángulo.

Capacitivos, térmicos y piezoeléctricos

Los capacitivos miden desplazamientos MEMS y humedad; NTC y PTC miden temperatura; el piezoeléctrico entrega carga con la fuerza dinámica (detonación). Las cámaras CMOS de alto rango dinámico ven a la vez el sol y la sombra.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Efecto Hall	$V_H = I \cdot B \,/\, (n \cdot q \cdot t)$	t: espesor de la placa.
Nibble SENT	$t = (12 + \text{valor}) \cdot \text{tick}$	Tick típico 3 µs.
Peor caso	$e = \Sigma \lvert e_i \rvert$	Conservador.
Estadístico (RSS)	$e = \sqrt{(\Sigma e_i^{\,2})}$	Errores independientes.

Ejemplo resuelto: una trama SENT de presión y temperatura

1. Señal 1 = 0x5A3, señal 2 = 0x1F0, estado 0.
2. Nibbles de datos: 5, A, 3, 1, F, 0; CRC-4 con semilla 5 = 7.
3. Ticks: 56, 12, 17, 22, 15, 13, 27, 12, 19 → 193 ticks = 579 µs.
4. Si un nibble cambia en el camino, el CRC no coincide y la trama se descarta.

El receptor recupera 0x5A3 y 0x1F0. Para el error del sensor, cuatro términos de 0,5 + 0,8 + 1,0 + 0,1 % suman 2,4 % en el peor caso y 1,38 % en RSS (sent_y_tolerancias.py).

Principios físicos y aplicaciones

Principio	Mide	Aplicación
Inductivo / Foucault	Posición, régimen	Cigüeñal, posición del pedal
Hall	Campo magnético	Árbol de levas, velocidad de rueda activa
AMR / GMR	Dirección del campo	Ángulo de dirección, posición
Capacitivo	Desplazamiento, humedad	Aceleración MEMS, humedad
Piezorresistivo	Deformación	Presión
Piezoeléctrico	Fuerza dinámica	Detonación

Código del capítulo

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

sent_y_tolerancias.py — codificación y decodificación SENT con CRC-4 y presupuesto de error peor caso y RSS. *(extracto: primeras 60 de 61 líneas)*

```
"""
sent_y_tolerancias.py — Capítulo 05: codificación y decodificación de una trama SENT
(SAE J2716) con su CRC-4 y presupuesto de error de un sensor.
"""
import math

TICK_US = 3.0


def tabla_crc4():
    """t[i] = i·x⁴ mod (x⁴ + x³ + x² + 1)."""
    t = []
```

```
    for i in range(16):
        v = i << 4
        for bit in range(7, 3, -1):
            if v & (1 << bit):
                v ^= 0b11101 << (bit - 4)
        t.append(v)
    return t


T = tabla_crc4()


def crc4(nibbles):
    crc = 5                      # semilla 0101
    for n in nibbles:
        crc = n ^ T[crc]
    return T[crc]                # aumentado con un nibble 0


def codificar(estado, senal1, senal2):
    datos = [(senal1 >> 8) & 0xF, (senal1 >> 4) & 0xF, senal1 & 0xF,
             (senal2 >> 8) & 0xF, (senal2 >> 4) & 0xF, senal2 & 0xF]
    nibbles = [estado] + datos + [crc4(datos)]
    return [56] + [12 + n for n in nibbles]          # duraciones en ticks


def decodificar(ticks):
    if ticks[0] != 56:
        raise ValueError('falta el pulso de sincronización')
    n = [t - 12 for t in ticks[1:]]
    datos, crc = n[1:7], n[7]
    if crc4(datos) != crc:
        return None
    return (datos[0] << 8 | datos[1] << 4 | datos[2], datos[3] << 8 | datos[4] << 4 | datos[5])


if __name__ == '__main__':
    print('Tabla CRC-4:', T)
    ticks = codificar(0, 0x5A3, 0x1F0)
    print('Ticks:', ticks)
    print(f'Duración de la trama: {sum(ticks)} ticks = {sum(ticks) * TICK_US:.0f} µs')
    print('Decodificada:', [hex(x) for x in decodificar(ticks)])
    ticks[3] += 1                                    # un nibble alterado
    print('Con un nibble alterado:', decodificar(ticks))

    errores = {'offset': 0.5, 'ganancia': 0.8, 'deriva térmica': 1.0, 'cuantización ADC': 0.1}
    peor = sum(errores.values())
    rss = math.sqrt(sum(e * e for e in errores.values()))
```

Errores frecuentes

- Sumar los errores en el peor caso y sobredimensionar el sensor (o hacer RSS con errores correlacionados).
- Olvidar el nibble cero de aumento en el CRC SENT.
- Pensar que un sensor Hall mide velocidad cero igual que un inductivo: el inductivo no genera señal parado.

Autoevaluación

1. ¿Qué protocolo usan los sensores de airbag?
2. ¿Cuánto dura el pulso de sincronización SENT?
3. ¿Qué sensor necesita la estructura barber pole?

Respuestas. 1. PSI5. **2.** 56 ticks. **3.** El AMR.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Sensores de posición, velocidad, presión, caudal y oxígeno

Capítulo 06 · En el video: **minuto 1:15:42** · Duración 15:07

Objetivo. Calcular la medida de los sensores de chasis y motor más usados a partir de su señal.

Ideas clave

Velocidad de rueda

El sensor pasivo (inductivo) entrega una tensión que crece con la velocidad y no mide bajo unos km/h; el activo (Hall o AMR) entrega pulsos de corriente desde cero y puede indicar el sentido de giro.

Pedal, par y ángulo de dirección

El pedal electrónico usa dos pistas redundantes. El ángulo de dirección (LWS) debe medir varias vueltas del volante: un nonio con dos piñones de distinto número de dientes resuelve la vuelta exacta.



Principio del sensor LWS del capítulo 06.

Acelerómetros y giróscopos MEMS

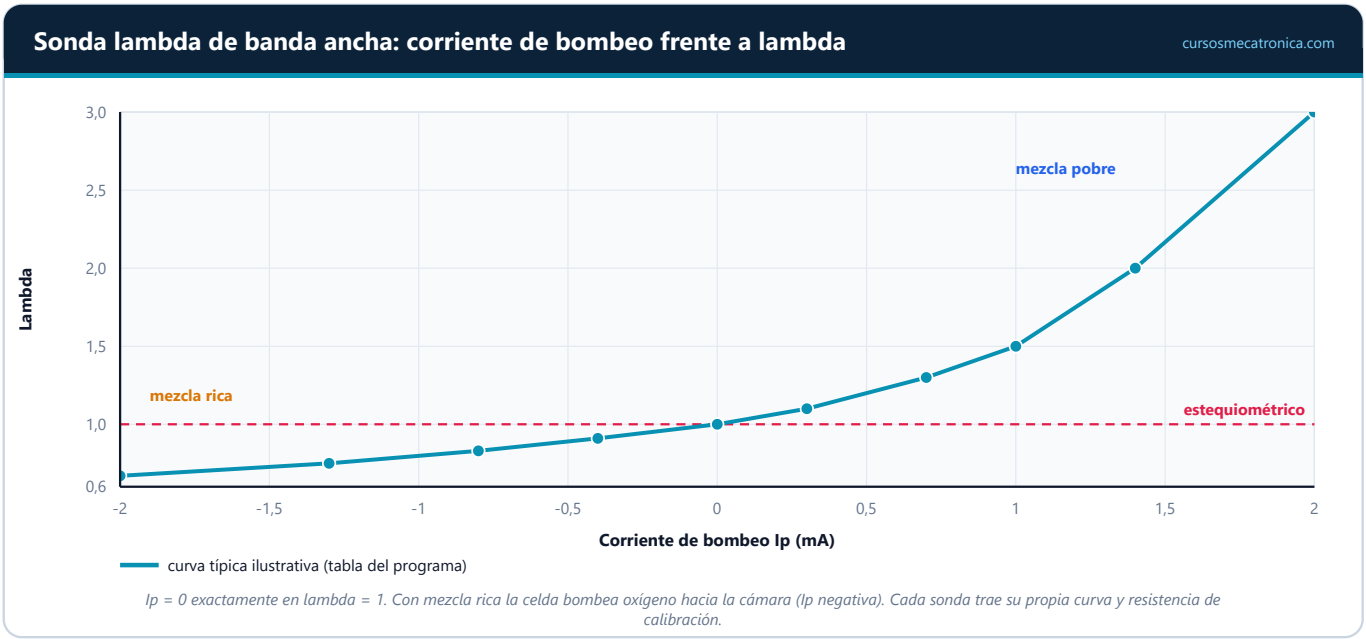
Una masa con peines interdigitados cambia su capacidad al acelerarse. El giróscopo hace vibrar una masa: al girar el vehículo, la fuerza de Coriolis la desplaza en perpendicular, proporcional a la velocidad de guiñada.

Presión, caudal y detonación

Membranas de silicio con piezorresistencias miden presión. El caudalímetro de película caliente (HFM) mantiene una resistencia sobre la temperatura del aire y mide la potencia que se lleva el flujo. El sensor de detonación piezoeléctrico escucha la vibración del bloque.

Sondas lambda

La binaria de circonio salta alrededor de lambda = 1. La de banda ancha (LSU) tiene una celda de bombeo: la corriente necesaria para mantener la celda de medida en lambda = 1 indica la mezcla de 0,65 a aire puro.



Curva usada en sensores_aplicados.py.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Velocidad de rueda	$v = (f / z)\cdot\pi\cdot D$	z: dientes de la rueda fónica.
Nonio	$\psi \approx (\varphi_1 - \varphi_2) / (r_1 - r_2)$	Luego se afina con φ_1.
Coriolis	$F = 2m\cdot v \times \Omega$	Base del giróscopo.
Película caliente	$P = (a + b\cdot\sqrt{\dot{m}})\cdot\Delta T$	Ley de King.

Ejemplo resuelto: velocidad y ángulo de dirección

1. Rueda fónica de 48 dientes, neumático de 0,63 m: a 1000 Hz,
   $v = 1000/48\cdot\pi\cdot 0{,}63 = 41{,}23\ m/s = 148{,}4\ km/h$.
2. Nonio 60/30/31: r_1 = 2, r_2 = 1,935; rango sin ambigüedad ±2790°.
3. Volante a 412,7°: los sensores leen 105,6° y 78,6° (con ±0,2° de error).
4. El algoritmo reconstruye 412,8°.

Todo el recorrido del volante (±780°) se mide sin contar vueltas desde el arranque. sensores_aplicados.py interpola también la curva de la sonda LSU (Ip = −0,6 mA → λ = 0,87).

Sensores aplicados

Sensor	Principio	Salida
Velocidad de rueda activo	Hall / AMR	Pulsos de corriente 7/14 mA
Ángulo de dirección	AMR + nonio	CAN
Aceleración y guiñada	MEMS capacitivo / Coriolis	SPI o CAN
Presión del colector	Piezorresistivo	Analógica o SENT
Masa de aire	Película caliente	Frecuencia
Lambda de banda ancha	Celda de bombeo de circonio	Corriente Ip

Código del capítulo

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

sensores_aplicados.py — velocidad de rueda, ángulo de dirección por nonio y lambda por interpolación de Ip.

```
"""
sensores_aplicados.py — Capítulo 06: velocidad de rueda, ángulo de dirección multivuelta
por nonio (vernier) y sonda lambda de banda ancha por interpolación.
"""
import math

Z_PRINCIPAL, Z1, Z2 = 60, 30, 31          # dientes de la corona y de los dos piñones


def velocidad_rueda(f_hz, dientes=48, diametro_m=0.63):
    return f_hz / dientes * math.pi * diametro_m


def lecturas(psi):
    """Ángulo medido (0–360°) por cada sensor AMR en su piñón."""
    return (psi * Z_PRINCIPAL / Z1) % 360, (psi * Z_PRINCIPAL / Z2) % 360


def vernier(phi1, phi2):
    r1, r2 = Z_PRINCIPAL / Z1, Z_PRINCIPAL / Z2
    delta = (phi1 - phi2) % 360
    psi_grueso = delta / (r1 - r2)
    if psi_grueso > 180 / (r1 - r2):            # rango centrado en cero
        psi_grueso -= 360 / (r1 - r2)
    k = round((psi_grueso * r1 - phi1) / 360)
    return (phi1 + 360 * k) / r1


# Curva típica (ilustrativa) corriente de bombeo -> lambda; usar la del fabricante
LSU = [(-2.0, 0.67), (-1.3, 0.75), (-0.8, 0.83), (-0.4, 0.91), (0.0, 1.00),
       (0.3, 1.10), (0.7, 1.30), (1.0, 1.50), (1.4, 2.00), (2.0, 3.00)]


def lambda_lsu(ip_ma):
    for (i0, l0), (i1, l1) in zip(LSU, LSU[1:]):
        if i0 <= ip_ma <= i1:
            return l0 + (l1 - l0) * (ip_ma - i0) / (i1 - i0)
    raise ValueError('fuera de la tabla')


if __name__ == '__main__':
    for f in (100, 1000, 2000):
```

```
        v = velocidad_rueda(f)
        print(f'Rueda fónica 48 dientes, {f:4d} Hz -> {v:5.2f} m/s = {3.6 * v:5.1f} km/h')

    print(f'\nNonio {Z_PRINCIPAL}/{Z1}/{Z2}: rango sin ambigüedad ±{180 / (Z_PRINCIPAL / Z1 - Z_PRINCIPAL /
Z2):.0f}°')
    for psi in (-780.0, -95.3, 0.0, 412.7, 780.0):
        p1, p2 = lecturas(psi)
        p1, p2 = p1 + 0.2, p2 - 0.2                 # error de medición ±0,2°
        print(f'  volante {psi:7.1f}° -> sensores {p1 % 360:6.1f}° / {p2 % 360:6.1f}° -> {vernier(p1 % 360, p2
% 360):7.1f}°')

    for ip in (-0.6, 0.0, 0.5):
        print(f'Sonda LSU, Ip = {ip:+.1f} mA -> lambda = {lambda_lsu(ip):.3f}')
```

Errores frecuentes

- Usar un sensor de rueda pasivo para funciones que necesitan velocidad cero (arranque en pendiente).
- Calibrar la sonda LSU sin la resistencia de compensación de su conector.
- Olvidar el rango del nonio y salir de él con desmultiplicaciones grandes.

Autoevaluación

1. ¿Qué fuerza aprovecha el giróscopo MEMS?
2. ¿Qué indica Ip = 0 en la sonda LSU?
3. ¿A qué velocidad corresponde 1000 Hz en el ejemplo?

Respuestas. 1. La de Coriolis. **2.** Lambda = 1. **3.** 148,4 km/h.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Actuadores eléctricos y electrohidráulicos: PWM y motores BLDC

Capítulo 07 · En el video: **minuto 1:30:49** · Duración 15:10

Objetivo. Elegir la etapa de potencia de un actuador y controlar la corriente de una electroválvula proporcional.

Ideas clave

Cadena de actuación

Señal de mando → etapa de potencia → convertidor de energía (motor, solenoide, piezo) → mecanismo. Los principios son el electrodinámico (fuerza de Lorentz sobre un conductor) y el electromagnético (atracción de un núcleo, tensión de Maxwell).

PWM y puente H

Conmutar a frecuencia fija y variar el ciclo de trabajo regula la tensión media con pocas pérdidas. Un interruptor de lado bajo o alto mueve en un sentido; el puente H permite invertir y frenar. El diodo de rueda libre protege al apagar una carga inductiva.

Motores

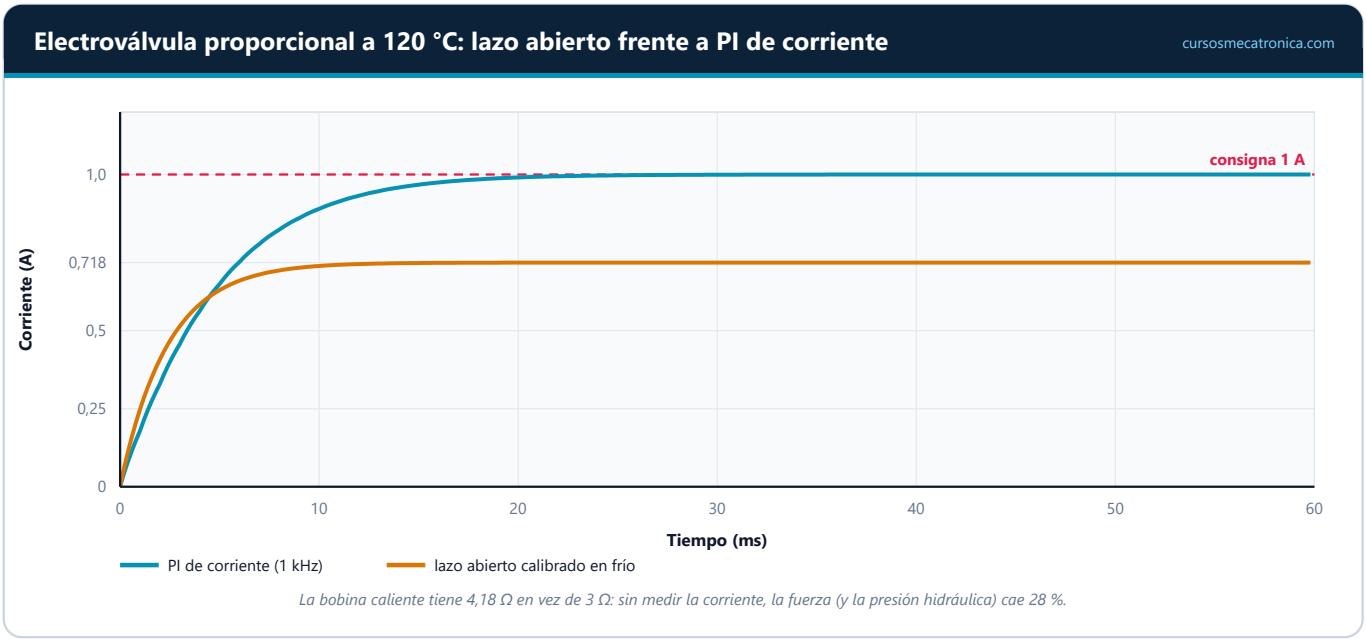
El DC con escobillas es barato (alzavidrios, limpiaparabrisas). El BLDC/PMSM conmutado electrónicamente dura más y es más eficiente (bombas, ventiladores, dirección). El paso a paso posiciona en lazo abierto (faros, mariposas antiguas).

Actuadores electrohidráulicos

En transmisiones automáticas y frenos se mueven válvulas con solenoides. La 3/2 de asiento es de todo o nada; las moduladas por PWM o los reguladores proporcionales de corredera fijan presión según la corriente.

Control de corriente y dither

La fuerza del solenoide depende de la corriente, no de la tensión. Como la resistencia de la bobina sube 0,39 % por °C, se mide la corriente y un PI ajusta el PWM. Un pequeño dither (oscilación de 100–300 Hz) evita que la corredera se pegue por fricción estática.



Salida de control_corriente_solenoide.py.

Diseño por simulación

Los modelos 1D multidominio (eléctrico, magnético, hidráulico) se combinan con dinámica de fluidos (CFD) y elementos finitos para la estructura y el circuito magnético.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Lorentz	$F = B \cdot I \cdot l$	Motor electrodinámico.
Tensión media	$V_{med} = d \cdot V_{bat}$	d: ciclo de trabajo.
Bobina	$L \cdot di/dt = d \cdot V - R(T) \cdot i$	$R(T) = R_{20}[1 + 0{,}00393(T - 20)]$.
Constante de tiempo	$\tau = L/R$	10 mH / 4,18 Ω = 2,4 ms.

Ejemplo resuelto: electroválvula caliente

1. Bobina de 3 Ω a 20 °C y 10 mH, batería 13,5 V; se pide 1 A.
2. A 120 °C, $R = 3 \cdot (1 + 0{,}00393 \cdot 100) = 4{,}18\ \Omega$ → hace falta 31,0 % de ciclo (22,2 % en frío).
3. Con el ciclo calibrado en frío la corriente queda en 0,718 A: 28 % menos fuerza.
4. El PI de corriente a 1 kHz llega a 1,000 A (95 % en 13,3 ms).

El lazo de corriente compensa la temperatura; con dither de ±0,1 A a 100 Hz la corriente oscila entre 0,969 y 1,031 A (control_corriente_solenoide.py).

Motores en el automóvil

Tipo	Ventaja	Uso
DC con escobillas	Barato y simple	Alzavidrios, asientos, limpiaparabrisas
BLDC / PMSM	Eficiente, sin desgaste	Bombas, ventiladores, EPS, tracción
Paso a paso	Posición en lazo abierto	Nivelación de faros, indicadores
Solenoide	Fuerza lineal rápida	Válvulas de transmisión, inyectores

Código del capítulo

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

control_corriente_solenoide.py — modelo PWM de una bobina, lazo abierto frente a PI de corriente y dither.

```
"""
control_corriente_solenoide.py — Capítulo 07: control PI de la corriente de una
electroválvula proporcional con PWM, dither y compensación de temperatura.
"""
import math

V_BAT, L, R20, ALFA = 13.5, 0.010, 3.0, 0.00393


def r_bobina(temp_c):
    return R20 * (1 + ALFA * (temp_c - 20))


def simular(i_ref, temp_c, lazo_cerrado=True, t_fin=0.2, dt=1e-5, dither=0.0):
    """Modelo promedio del PWM: L·di/dt = d·V − R·i. Control a 1 kHz."""
    r = r_bobina(temp_c)
    i, integ, duty = 0.0, 0.0, 0.0
    kp, ki = 0.10, 60.0
    paso_control = int(1e-3 / dt)
    hist = []
    for k in range(int(t_fin / dt)):
        t = k * dt
        if k % paso_control == 0:
            ref = i_ref + dither * math.sin(2 * math.pi * 100 * t)
            if lazo_cerrado:
                e = ref - i
                integ += ki * e * 1e-3
                duty = min(1.0, max(0.0, kp * e + integ))
            else:
                duty = ref * r_bobina(20) / V_BAT      # calibrado en frío
        i += (duty * V_BAT - r * i) / L * dt
        hist.append((t, i))
    return i, duty, hist


if __name__ == '__main__':
    for t in (20, 120):
        print(f'Bobina a {t:3d} °C: R = {r_bobina(t):.2f} Ω,'
              f' ciclo de trabajo para 1 A = {100 * r_bobina(t) / V_BAT:.1f} %')
    i_abierto, _, _ = simular(1.0, 120, lazo_cerrado=False)
    print(f'\nLazo abierto calibrado a 20 °C, bobina a 120 °C: {i_abierto:.3f} A'
          f' ({100 * (i_abierto - 1):.0f} %)')
    i_pi, d, hist = simular(1.0, 120)
    t95 = next(t for t, i in hist if i >= 0.95)
    print(f'PI de corriente a 120 °C: {i_pi:.3f} A con {100 * d:.1f} % de ciclo; 95 % en {1000 * t95:.1f} ms')
    _, _, hd = simular(1.0, 120, dither=0.1)
```

```
    ult = [i for t, i in hd if t > 0.15]
    print(f'Con dither de ±0,1 A a 100 Hz: la corriente oscila entre {min(ult):.3f} y {max(ult):.3f} A')
```

Errores frecuentes

- Controlar una válvula proporcional por tensión: la fuerza deriva con la temperatura.
- Omitir el diodo de rueda libre y destruir el transistor al apagar.
- Dither demasiado grande: la presión oscila en vez de estabilizarse.

Autoevaluación

1. ¿Por qué se controla la corriente y no la tensión de un solenoide?
2. ¿Para qué sirve el dither?
3. ¿Qué permite el puente H?

Respuestas. 1. Porque la fuerza depende de la corriente y la resistencia cambia con la temperatura. **2.** Para vencer la fricción estática de la corredera. **3.** Invertir el sentido de giro y frenar.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Control electrónico de la transmisión: AT, DCT, CVT, DHT y TCU

Capítulo 08 · En el video: **minuto 1:45:59** · Duración 14:39

Objetivo. Explicar cómo una transmisión automática cambia de marcha sin tirones y calcular relaciones y energía de un cambio.

Ideas clave

Gestión coordinada

La TCU decide la marcha según pedal, velocidad y carga, y coordina con el motor para que el cambio sea suave. Las transmisiones manuales automatizadas (AST) añaden actuadores al embrague y la palanca.

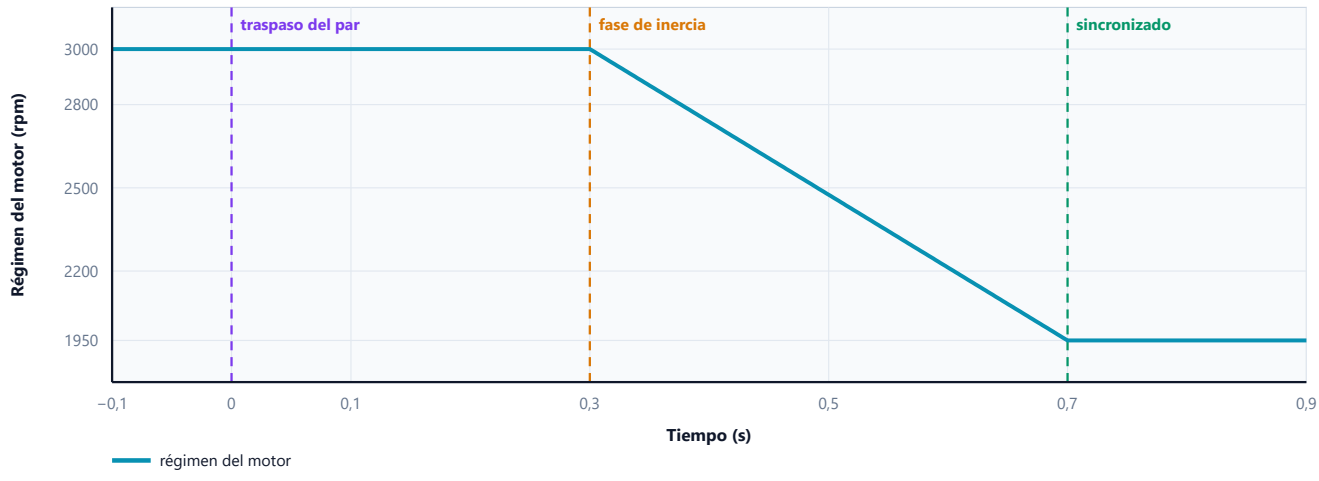
Convertidor de par y planetarios

El convertidor hidrodinámico multiplica el par al arrancar y se bloquea con un embrague a velocidad de crucero. Los trenes planetarios dan varias relaciones según qué elemento se frena o se une.

Cambio con solape

Un embrague sale mientras otro entra. En la fase de traspaso el par pasa de uno a otro; en la fase de inercia el entrante frena el motor hasta el nuevo régimen. La presión se adapta durante la vida de la caja para compensar el desgaste.

Cambio ascendente 2,0 → 1,3 con solape de embragues (250 N·m) cursosmecatronica.com



En el traspaso el embrague entrante toma el par sin cambiar el régimen; en la fase de inercia frena el motor hasta 1950 rpm. Se disipan 10,4 kJ.

Salida de cambio_con_solape.py.

Intervención en el par motor

Reducir el par del motor durante la fase de inercia acorta el cambio y baja la energía disipada en el embrague (y su temperatura).

DCT, CVT y DHT

La de doble embrague (DCT) preselecciona la siguiente marcha en el otro eje; la CVT varía la relación de forma continua con poleas y correa de empuje; la DHT integra motores eléctricos en la transmisión de los híbridos.

TCU integrada

La unidad de control va sumergida en aceite (ATF) a más de 120 °C, sobre sustratos cerámicos LTCC o PCB de alta temperatura; la temperatura de unión de sus semiconductores limita la potencia que puede manejar.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Willis (corona fija)	$i = 1 + Z_r/Z_s$	Entra sol, sale portasatélites.
Willis (portasatélites fijo)	$i = -Z_r/Z_s$	Marcha atrás.
Energía en el embrague	$E = \int T \cdot \Delta\omega \, dt$	Δω: deslizamiento.
Temperatura de unión	$T_j = T_{aceite} + P \cdot R_{th}$	Límite típico 150–175 °C.

Ejemplo resuelto: cambio de 2,0 a 1,3

1. Motor a 3000 rpm con 250 N·m; tras el cambio baja a $3000 \cdot 1{,}3/2{,}0 = 1950$ rpm.
2. Traspaso de 0,3 s: el embrague entrante sube de 0 a 250 N·m mientras desliza.
3. Inercia de 0,4 s con 15 % de par extra para frenar el motor.
4. Integrando par × deslizamiento: $E = 10{,}4$ kJ.

Si la fase de inercia dura 0,8 s el cambio es más suave pero se disipan 16,8 kJ. Con 2 W en la TCU y 15 K/W: T_j = 150 °C con aceite a 120 °C (cambio_con_solape.py).

Tipos de transmisión automática

Tipo	Principio	Ventaja	Cuidado
AT	Convertidor + planetarios	Suavidad, par alto	Pérdidas del convertidor
DCT	Dos embragues y dos ejes	Cambios muy rápidos	Arrancadas y calor en embragues
CVT	Poleas variables y correa	Régimen óptimo continuo	Par limitado
DHT	Motores eléctricos integrados	Hibridación eficiente	Complejidad de control

Código del capítulo

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

cambio_con_solape.py — relaciones de Willis, energía disipada en un cambio con solape y temperatura de unión de la TCU.

```
"""
cambio_con_solape.py — Capítulo 08: relaciones de un tren planetario (Willis), energía
disipada en un cambio con solape y temperatura de unión de la TCU.
"""
```

```
def willis(zs, zr, fijo):
    """Relación entrada/salida de un planetario simple según el elemento fijo."""
    if fijo == 'corona':        # entra sol, sale portasatélites
        return 1 + zr / zs
    if fijo == 'sol':           # entra corona, sale portasatélites
        return 1 + zs / zr
    if fijo == 'portasatelites':  # entra sol, sale corona (invierte)
        return -zr / zs
    raise ValueError(fijo)


def cambio(t_motor=250.0, n_ini=3000.0, i1=2.0, i2=1.3, t_traspaso=0.3, t_inercia=0.4, dt=0.001):
    """Cambio ascendente: fase de traspaso del par y fase de inercia (sincronización)."""
    n_fin = n_ini * i2 / i1
    energia, t = 0.0, 0.0
    w_sal = n_ini / i1 * 2 * 3.14159265 / 60        # rad/s del eje de salida (constante)
    while t < t_traspaso + t_inercia:
        if t < t_traspaso:
            par_b = t_motor * t / t_traspaso          # el embrague entrante toma el par
            n_motor = n_ini
        else:
            par_b = t_motor * 1.15                    # exceso para frenar el motor
            n_motor = n_ini + (n_fin - n_ini) * (t - t_traspaso) / t_inercia
        w_motor = n_motor * 2 * 3.14159265 / 60
        deslizamiento = w_motor - w_sal * i2          # velocidad relativa en el embrague B
        energia += par_b * deslizamiento * dt
        t += dt
    return n_fin, energia


if __name__ == '__main__':
    zs, zr = 30, 72
    for fijo in ('corona', 'sol', 'portasatelites'):
        print(f'Planetario {zs}/{zr}, {fijo} fija: i = {willis(zs, zr, fijo):+.3f}')
    n_fin, e = cambio()
    print(f'\nCambio 2,0 -> 1,3 a 3000 rpm con 250 N·m: el motor baja a {n_fin:.0f} rpm;'
          f' energía en el embrague = {e / 1000:.1f} kJ')
    _, e2 = cambio(t_inercia=0.8)
    print(f'Con fase de inercia de 0,8 s: {e2 / 1000:.1f} kJ (más suave, pero más calor)')
    t_aceite, p, rth = 120.0, 2.0, 15.0
    print(f'\nTCU sumergida: Tj = {t_aceite:.0f} + {p} W × {rth} K/W = {t_aceite + p * rth:.0f} °C (límite 175
°C)')
```

Errores frecuentes

- Soltar el embrague saliente antes de que el entrante tome el par (el motor se embala).
- Alargar la fase de inercia sin vigilar el calor del embrague.
- Olvidar que la TCU trabaja dentro del aceite caliente.

Autoevaluación

1. ¿Qué relación da un planetario 30/72 con la corona fija?
2. ¿Qué sucede en la fase de inercia?
3. ¿Qué preselecciona una DCT?

Respuestas. 1. 3,4. **2.** El embrague entrante frena el motor al nuevo régimen. **3.** La siguiente marcha en el otro eje.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Frenos ABS, deslizamiento, frenado regenerativo y control de tracción

Capítulo 09 · En el video: **minuto 2:00:38** · Duración 14:17

Objetivo. Explicar el lazo de regulación del ABS y estimar su efecto en la distancia de frenado.

Ideas clave

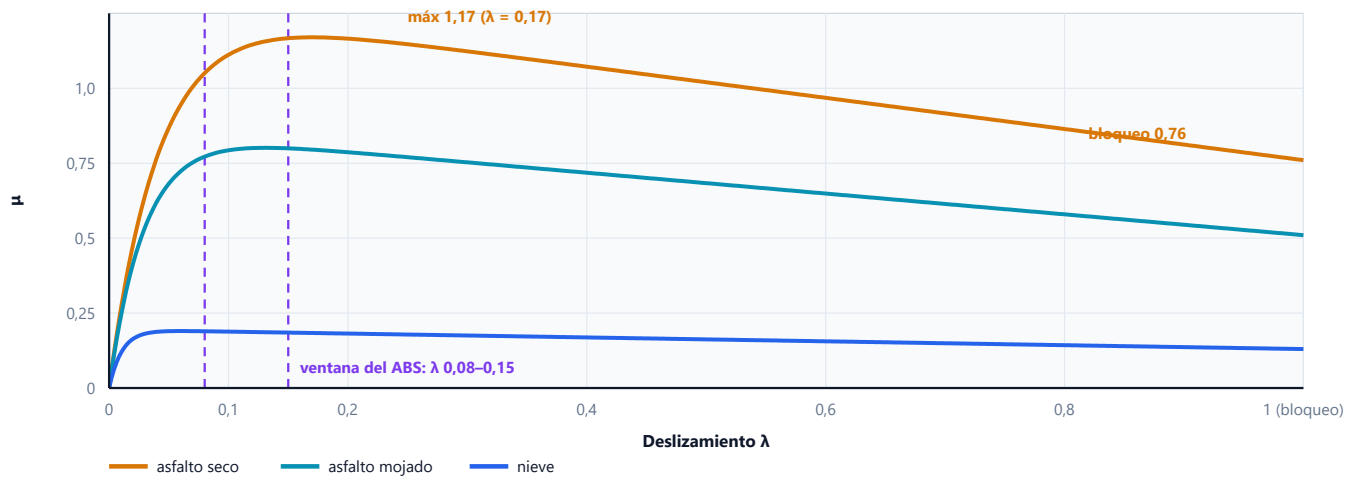
Arquitectura del ABS

Sensores de velocidad en cada rueda, una unidad hidráulica con válvulas de admisión y escape por rueda, una bomba de retorno y la ECU. Si el sistema falla, el freno convencional sigue funcionando.

Contacto neumático-calzada

La fuerza de frenado depende del deslizamiento $\lambda = (v - \omega \cdot r)/v$. La adherencia crece hasta un máximo ($\lambda \approx$ 0,1–0,2 en asfalto) y luego cae; con la rueda bloqueada además se pierde la capacidad de girar.

Coeficiente de adherencia según el deslizamiento (modelo de Burckhardt) cursosmecatronica.com



Al bloquear la rueda se pierde adherencia longitudinal y casi toda la lateral (no se puede girar). El ABS mantiene λ cerca del máximo.

Curvas usadas en abs_cuarto_de_vehiculo.py.

Ciclos de regulación

Si la rueda se desacelera demasiado o λ supera un umbral, la ECU mantiene y luego baja la presión; cuando la rueda recupera velocidad la vuelve a subir en escalones. En baja adherencia los ciclos son más lentos.

Calzada asimétrica

Con un lado en hielo y otro en asfalto, frenar a fondo crea un momento de guiñada. La atenuación (GMA) sube la presión del lado de alta adherencia de forma gradual para que el conductor alcance a corregir.

Frenado regenerativo

En eléctricos e híbridos el motor frena recuperando energía hasta su límite de potencia; el resto lo pone el freno hidráulico (brake blending) sin que el conductor note el cambio en el pedal.

Control de tracción

El TCS reduce el par motor y frena la rueda que patina; el diferencial electrónico (EDS) frena una rueda para pasar par a la otra.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Deslizamiento al frenar	$\lambda = (v - \omega \cdot r)/v$	0 libre, 1 bloqueada.
Burckhardt	$\mu(\lambda) = c_1(1 - e^{-c_2\lambda}) - c_3\lambda$	Parámetros según la calzada.
Rueda	$J \cdot \dot{\omega} = F_x \cdot r - T_{freno}$	$F_x = \mu \cdot N$.
Reparto regenerativo	$F_{hidr} = F_{total} - \min(F_{total}, F_{regen,máx})$	Brake blending.

Ejemplo resuelto: frenado de emergencia desde 100 km/h

1. Cuarto de vehículo de 400 kg, rueda de 0,30 m, par de freno hasta 2500 N·m.
2. Sin ABS la rueda se bloquea y trabaja con μ = 0,76 (seco): 51,5 m.
3. Con ABS (λ entre 0,08 y 0,15) se aprovecha cerca del máximo de 1,17: 36,8 m.
4. En mojado: 76,5 m sin ABS y 50,9 m con ABS.

Además de acortar la distancia, el ABS conserva la dirección. Un frenado de 4 m/s² en un eléctrico de 1600 kg se reparte en 4,0 kN regenerativos y 2,4 kN hidráulicos (abs_cuarto_de_vehiculo.py).

Funciones de dinámica longitudinal

Función	Actúa sobre	Objetivo
ABS	Presión de freno por rueda	Evitar el bloqueo
GMA	Subida de presión asimétrica	Limitar la guiñada en μ partido
Brake blending	Motor eléctrico + hidráulica	Recuperar energía
TCS	Par motor + freno	Evitar el patinamiento
EDS	Freno de una rueda	Pasar par a la otra rueda

Código del capítulo

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

abs_cuarto_de_vehiculo.py — curvas μ–λ, simulación de frenado con y sin ABS y reparto regenerativo.

```
"""
abs_cuarto_de_vehiculo.py — Capítulo 09: curvas μ-deslizamiento (Burckhardt), frenado de un
cuarto de vehículo con y sin ABS y reparto de frenado regenerativo.
"""
import math

SUPERFICIES = {'asfalto seco': (1.2801, 23.99, 0.52), 'asfalto mojado': (0.857, 33.822, 0.347),
               'nieve': (0.1946, 94.129, 0.0646)}
M, J, R, G = 400.0, 1.2, 0.30, 9.81
```

```
def mu(lam, sup):
    c1, c2, c3 = SUPERFICIES[sup]
    return c1 * (1 - math.exp(-c2 * lam)) - c3 * lam


def optimo(sup):
    mejor = max(range(1, 1001), key=lambda k: mu(k / 1000, sup))
    return mejor / 1000, mu(mejor / 1000, sup)


def frenar(sup, abs_activo, v0=27.78, dt=1e-4):
    v, w, x, par = v0, v0 / R, 0.0, 0.0
    while v > 0.5:
        lam = max(0.0, (v - w * R) / v)
        if abs_activo:
            if lam > 0.15:
                par = max(0.0, par - 30000 * dt)       # baja la presión
            elif lam < 0.08:
                par = min(2500.0, par + 15000 * dt)   # sube la presión
        else:
            par = min(2500.0, par + 15000 * dt)
        fx = mu(lam, sup) * M * G
        v -= fx / M * dt
        w = max(0.0, w + (fx * R - par) / J * dt)
        x += v * dt
    return x


if __name__ == '__main__':
    for s in SUPERFICIES:
        lo, mo = optimo(s)
        print(f'{s:<15s}: μ máx {mo:.3f} con λ = {lo:.3f}; rueda bloqueada μ = {mu(1.0, s):.3f}')
    print('\nFrenado desde 100 km/h (cuarto de vehículo de 400 kg)')
    for s in ('asfalto seco', 'asfalto mojado'):
        print(f'  {s:<15s}: sin ABS {frenar(s, False):5.1f} m, con ABS {frenar(s, True):5.1f} m')

    pedido = 4.0                                       # m/s² pedidos por el conductor
    regen_max = 1600 * 2.5                             # N que admite el motor (≈ 2,5 m/s²)
    f_total = 1600 * pedido
    print(f'\nFrenado de 4 m/s²: regenerativo {min(f_total, regen_max) / 1000:.1f} kN,'
          f' hidráulico {max(0, f_total - regen_max) / 1000:.1f} kN')
```

Errores frecuentes

- Creer que el ABS siempre acorta la distancia: en grava suelta puede alargarla.
- Olvidar que la rueda bloqueada no genera fuerza lateral.
- Repartir el frenado regenerativo sin considerar el estado de carga de la batería.

Autoevaluación

1. ¿Qué deslizamiento da la máxima adherencia en asfalto seco según el modelo?
2. ¿Qué hace la GMA?
3. ¿Qué es el brake blending?

Respuestas. 1. $\lambda \approx 0{,}17$. **2.** Sube gradualmente la presión del lado de alta adherencia. **3.** El reparto entre freno regenerativo e hidráulico.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Control de estabilidad, ADAS y frenos by-wire

Capítulo 10 · En el video: **minuto 2:14:55** · Duración 14:24

Objetivo. Calcular la guiñada de referencia del ESP y decidir qué rueda frenar ante un subviraje o sobreviraje.

Ideas clave

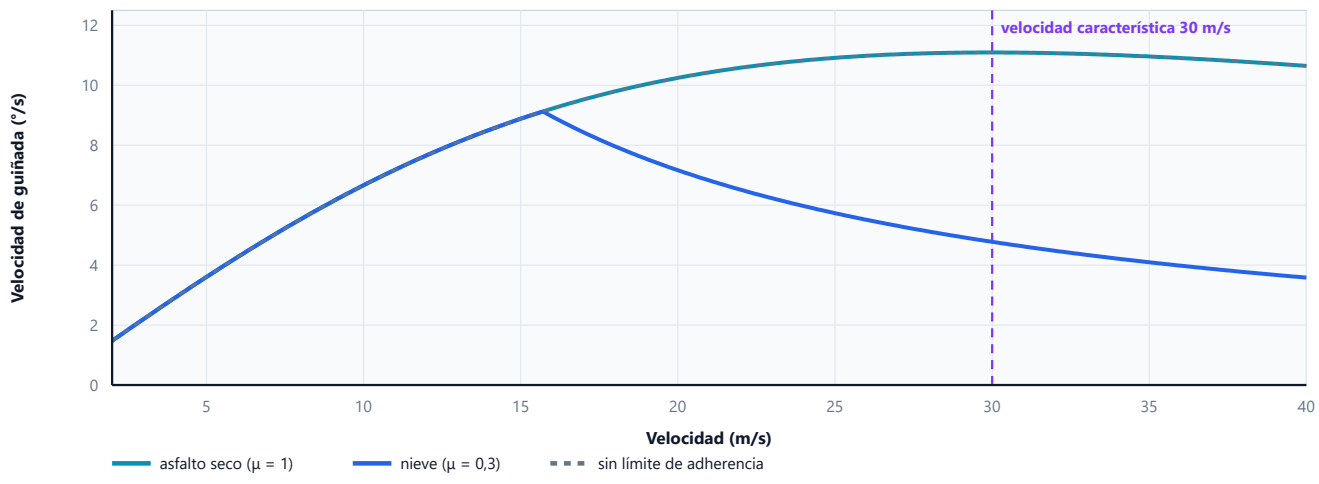
Subviraje y sobreviraje

En subviraje el auto gira menos de lo pedido (se va de frente); en sobreviraje gira más (la cola se abre). El ESP compara la guiñada medida con la de referencia y frena una sola rueda para crear un momento corrector.

Modelo de vía única

Junta las dos ruedas de cada eje en una. Con rigideces de deriva C_f y C_r, la guiñada estacionaria es $\psi = v \cdot \delta / [L(1 + K \cdot v^2)]$; K > 0 es un vehículo subvirador y $\sqrt{(1/K)}$ su velocidad característica.

Guiñada de referencia del ESP con 2° de ángulo en las ruedas cursosmecatronica.com

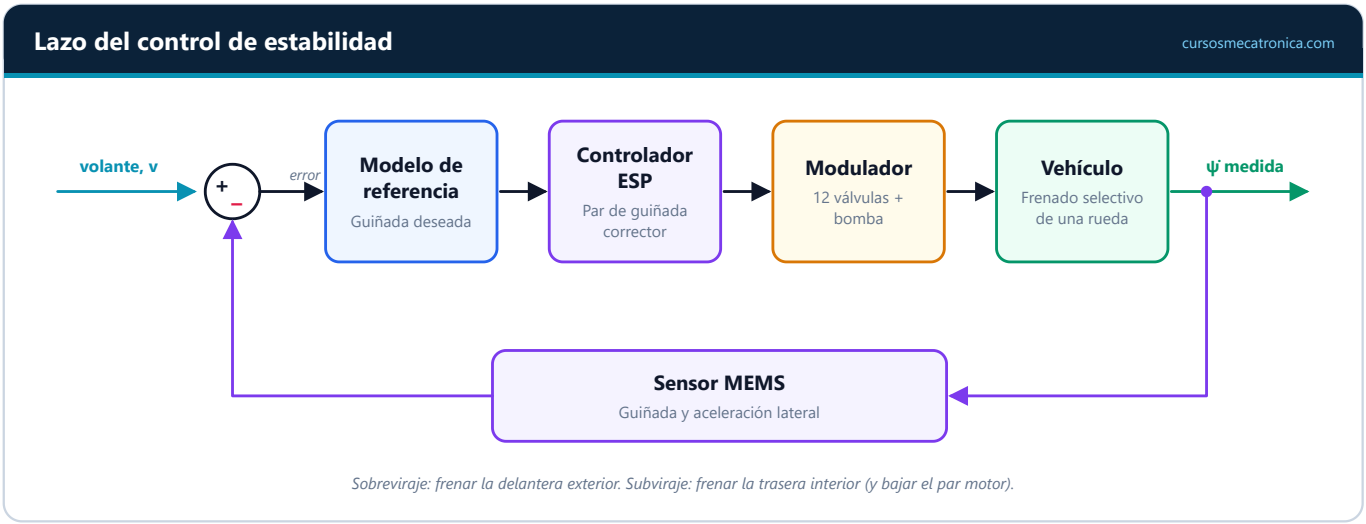


La referencia es la menor entre el modelo lineal y lo que permite la adherencia (0,85·μ·g/v). En nieve el auto no puede seguir la guiñada del modelo.

Salida de esp_modelo_de_bicicleta.py.

Lazo en cascada

Sensores MEMS de guiñada y aceleración lateral, ángulo de volante y velocidades de rueda alimentan un controlador que pide un par de guiñada; el modulador lo convierte en presiones por rueda y en una reducción del par motor.



Estructura del ESP del capítulo 10.

Reparto de frenado (EBD)

Al frenar, la carga se traslada al eje delantero. El EBD limita la presión trasera para que esas ruedas no se bloqueen primero.

Asistencias y ADAS

Asistente de frenado de emergencia, frenado autónomo (AEB) con radar y cámara, retención en pendiente y prellenado de los frenos. El modulador ESP de 12 válvulas y bomba autoaspirante permite frenar sin que el conductor pise.

Brake-by-wire

Servofrenos electromecánicos (iBooster) y sistemas "one-box" integran servo, ABS y ESP; con el pedal desacoplado, el frenado regenerativo y la conducción automatizada se coordinan mejor. La redundancia es obligatoria.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Guiñada estacionaria	$\dot{\psi} = v \cdot \delta \,/\, [L(1 + K \cdot v^2)]$	$L = l_f + l_r$.
Gradiente de subviraje	$K = m/L^2 \cdot (l_r/C_f - l_f/C_r)$	s^2/m^2.
Límite por adherencia	$\dot{\psi} \leq 0{,}85 \cdot \mu \cdot g \,/\, v$	$a_y = v \cdot \dot{\psi}$.
Transferencia de carga	$\Delta N = m \cdot a \cdot h \,/\, L$	h: altura del centro de gravedad.

Ejemplo resuelto: qué rueda frenar a 25 m/s

1. m = 1500 kg, l_f = 1,2 m, l_r = 1,5 m, C_f = 80 kN/rad, C_r = 90 kN/rad: K = 0,00111 s^2/m^2 (velocidad característica 30 m/s).
2. Con δ = 2° a 25 m/s: $\psi_{ref} = 10{,}92$ °/s.
3. Medida 12,6 °/s (más de 10 % sobre la referencia): sobreviraje → frenar la delantera exterior.
4. Medida 7,6 °/s: subviraje → frenar la trasera interior; 10,5 °/s: estable.

En nieve (μ = 0,3) la referencia se recorta a 4,78 °/s a 30 m/s: el ESP no pide lo imposible (esp_modelo_de_bicicleta.py).

Sistemas de freno electrónico

Sistema	Qué hace	Ejemplo de uso
ESP	Frena ruedas por separado y baja el par	Esquivar un obstáculo
EBD	Limita la presión trasera	Frenado fuerte cargado
AEB	Frena solo ante un choque inminente	Peatón que cruza
iBooster	Servo electromecánico	Brake blending, ADAS
One-box	Servo + ABS + ESP integrados	Vehículos eléctricos

Código del capítulo

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

esp_modelo_de_bicicleta.py — gradiente de subviraje, guiñada de referencia con límite de adherencia y decisión de frenado.

```
"""
esp_modelo_de_bicicleta.py — Capítulo 10: guiñada de referencia del modelo lineal de vía
única, límite por adherencia y decisión de frenado selectivo del ESP.
"""
import math

M, LF, LR, CF, CR = 1500.0, 1.2, 1.5, 80000.0, 90000.0
L = LF + LR
K = M / L ** 2 * (LR / CF - LF / CR)        # gradiente de subviraje (s²/m²)


def guinada_ref(v, delta_rad, mu=1.0):
    estacionaria = v * delta_rad / (L * (1 + K * v * v))
    limite = 0.85 * mu * 9.81 / v
    return math.copysign(min(abs(estacionaria), limite), delta_rad)


def decidir(medida, referencia, banda=0.1):
    if abs(medida) > abs(referencia) * (1 + banda):
        return 'SOBREVIRAJE: frenar la rueda delantera exterior'
    if abs(medida) < abs(referencia) * (1 - banda):
        return 'SUBVIRAJE: frenar la rueda trasera interior'
    return 'estable: sin intervención'


if __name__ == '__main__':
    print(f'K = {K:.5f} s²/m² -> velocidad característica {math.sqrt(1 / K):.1f} m/s'
```

```
          f' ({3.6 * math.sqrt(1 / K):.0f} km/h)')
d = math.radians(2.0)
print('\nVolante con 2° en la rueda: guiñada de referencia (°/s)')
for v in (10, 20, 30, 40):
    print(f'  {v:2d} m/s: seco {math.degrees(guinada_ref(v, d)):5.2f}'
          f'   nieve (μ = 0,3) {math.degrees(guinada_ref(v, d, 0.3)):5.2f}')
ref = guinada_ref(25, d)
print(f'\nA 25 m/s la referencia es {math.degrees(ref):.2f} °/s')
for medida in (12.6, 10.5, 7.6):
    print(f'  medida {medida:4.1f} °/s -> {decidir(math.radians(medida), ref)}')
```

Errores frecuentes

- Frenar la rueda equivocada: en sobreviraje es la delantera exterior.
- Usar la guiñada del modelo sin limitarla por la adherencia.
- Olvidar la transferencia de carga al repartir el frenado.

Autoevaluación

1. ¿Qué rueda frena el ESP en subviraje?
2. ¿Qué indica K > 0?
3. ¿Qué limita el EBD?

Respuestas. 1. La trasera interior. **2.** Un vehículo subvirador. **3.** La presión de las ruedas traseras.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Inyección diésel common rail: bombas e inyectores

Capítulo 11 · En el video: **minuto 2:29:19** · Duración 14:32

Objetivo. Relacionar la presión del riel con la masa inyectada y explicar cómo se logran inyecciones múltiples.

Ideas clave

Desacoplar presión e inyección

En el common rail la bomba mantiene un riel a presión (hasta 2500–3000 bar) independiente del régimen; la ECU decide cuándo y cuánto inyecta cada inyector.

Sistema common rail — cursosmecatronica.com



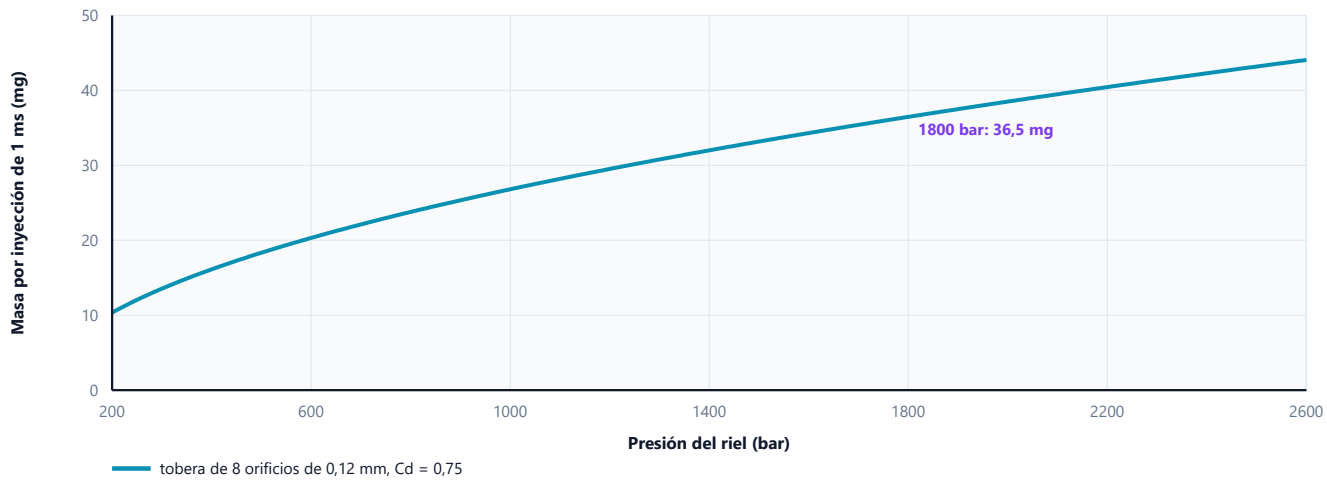
Generar la presión (bomba) y decidir la inyección (inyector) quedan desacoplados: la presión no depende del régimen.

Componentes del capítulo 11.

Presión y caudal

El riel es un acumulador: su presión sube con lo que entra y baja con lo que sale, según el módulo de compresibilidad del combustible. El caudal por la tobera crece con la raíz de la diferencia de presión.

Masa inyectada en 1 ms según la presión del riel — cursosmecatronica.com



El caudal crece con la raíz de la presión. Más presión atomiza mejor y permite inyectar lo mismo en menos tiempo.

Salida de riel_comun.py.

Inyección múltiple

La preinyección (1–2 mg) suaviza el ruido de combustión; la principal entrega el par; las postinyecciones queman hollín o calientan el catalizador y el filtro de partículas.

Bombas y riel

Generaciones CP1, CP3 y CP4 con dosificación en la entrada para no comprimir combustible que no se usa. El riel forjado lleva el sensor de presión (RDS) y una válvula reguladora. Una partícula de pocos micrómetros puede dañar un inyector: la limpieza es crítica.

Inyectores

En el de solenoide, una servoválvula descarga la cámara de control y la aguja sube por diferencia de presión. El piezoeléctrico mueve la servoválvula más rápido, con un acoplador hidráulico que amplifica la carrera; permite más inyecciones por ciclo y cantidades más pequeñas.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Presión del riel	$dp/dt = (E/V)\cdot(Q_{bomba} - Q_{iny} - Q_{fugas})$	E ≈ 1,6 GPa.
Caudal por la tobera	$Q = C_d \cdot A \cdot \sqrt{(2\Delta p/\rho)}$	Bernoulli con pérdidas.
Masa inyectada	$m = \rho \cdot Q \cdot t_{iny}$	Aproximación de aguja abierta.
Energía del piezo	$E = ½ \cdot C \cdot V^2$	Por cada carga.

Ejemplo resuelto: cuánto inyecta 1 ms a 1800 bar

1. Tobera de 8 orificios de 0,12 mm, C_d = 0,75, diésel de 830 kg/m³, cámara a 60 bar.
2. $Q = 0{,}75 \cdot 9{,}05 \cdot 10^{-8} \cdot \sqrt{(2 \cdot 1{,}74 \cdot 10^8 / 830)} = 43{,}9\ cm^3/s$.
3. En 1 ms: 36,5 mg. A 300 bar solo 13,5 mg.
4. Preinyección de 0,05 ms: 1,8 mg; principal de 0,65 ms: 23,7 mg.

Subir el riel de 800 a 1800 bar con la bomba a pleno toma unos 44 ms. Un piezo de 3 µF a 160 V gasta 38,4 mJ por carga (riel_comun.py).

Inyectores de common rail

	Solenoide	Piezoeléctrico
Accionamiento	Electroimán sobre servoválvula	Apilamiento piezo + acoplador hidráulico
Velocidad	Rápida	Muy rápida
Inyecciones por ciclo	Hasta 5	Hasta 8 o más
Electrónica	Corriente pico y mantención	Carga y descarga de un condensador a ~150 V

Código del capítulo

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

riel_comun.py — caudal y masa inyectada según presión, subida de presión del riel y energía del piezo.

```
"""
riel_comun.py — Capítulo 11: dinámica de presión del riel, caudal por la tobera,
masa inyectada y carrera del actuador piezoeléctrico.
"""
import math

E_DIESEL = 1.6e9      # Pa, módulo de compresibilidad a alta presión
V_RIEL = 25e-6        # m³
RHO = 830.0           # kg/m³


def caudal_tobera(dp_pa, orificios=8, diametro_mm=0.12, cd=0.75):
    area = orificios * math.pi * (diametro_mm * 1e-3) ** 2 / 4
    return cd * area * math.sqrt(2 * dp_pa / RHO)


def masa_inyectada(p_riel_bar, t_ms, p_camara_bar=60):
    q = caudal_tobera((p_riel_bar - p_camara_bar) * 1e5)
    return RHO * q * t_ms * 1e-3 * 1e6              # mg


def subir_presion(p0_bar, p_obj_bar, q_bomba_cm3s=40.0, q_fugas_cm3s=4.0, dt=1e-4):
    p, t = p0_bar * 1e5, 0.0
    while p < p_obj_bar * 1e5:
        p += E_DIESEL / V_RIEL * (q_bomba_cm3s - q_fugas_cm3s) * 1e-6 * dt
        t += dt
    return t


if __name__ == '__main__':
    for p in (300, 1000, 1800, 2500):
        print(f'Riel {p:4d} bar: caudal {caudal_tobera((p - 60) * 1e5) * 1e6:5.1f} cm³/s,'
              f' 1 ms inyecta {masa_inyectada(p, 1.0):5.1f} mg')
    print(f'\nSubir de 800 a 1800 bar con la bomba a pleno: {1000 * subir_presion(800, 1800):.0f} ms')
    pre, principal = masa_inyectada(1800, 0.05), masa_inyectada(1800, 0.65)
    print(f'Preinyección 0,05 ms = {pre:.1f} mg; principal 0,65 ms = {principal:.1f} mg')
    c, v = 3.0e-6, 160.0
    e = 0.5 * c * v * v
    print(f'\nPiezo de 3 µF a 160 V: {1000 * e:.1f} mJ por carga; con 2 inyecciones por ciclo,'
          f' 4 cilindros y 3000 rpm (25 ciclos/s): {e * 2 * 4 * 25:.1f} W')
```

Errores frecuentes

- Abrir el circuito de alta presión con el motor recién detenido.
- Contaminar los inyectores al cambiar el filtro.
- Suponer que la masa inyectada es proporcional a la presión (es a su raíz).

Autoevaluación

1. ¿Qué separa el common rail?
2. ¿Para qué sirve la preinyección?
3. ¿Qué ventaja da el piezo?

Respuestas. 1. La generación de presión de la inyección. **2.** Para suavizar la combustión y bajar el ruido. **3.** Más rapidez y más inyecciones pequeñas por ciclo.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Control electrónico diésel y postratamiento de emisiones

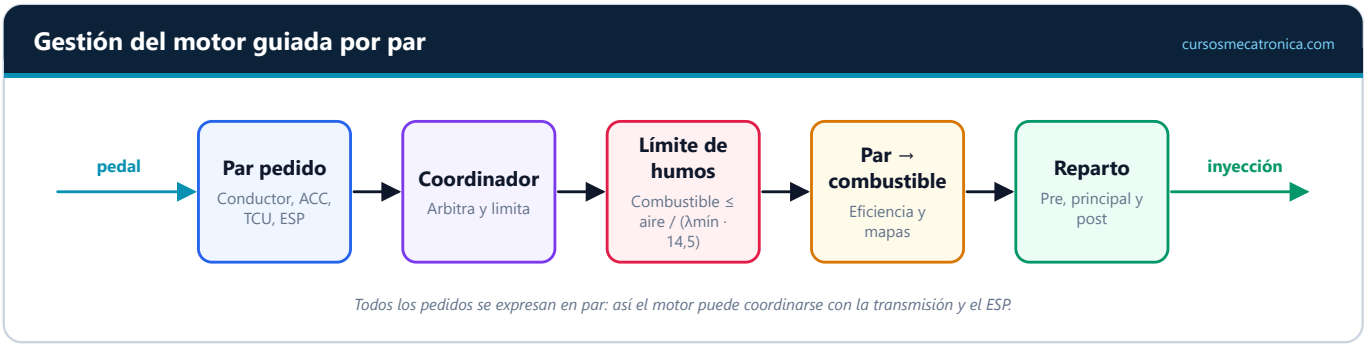
Capítulo 12 · En el video: **minuto 2:43:51** · Duración 14:23

Objetivo. Seguir el camino de un pedido de par hasta la inyección y estimar el consumo de AdBlue.

Ideas clave

Gestión guiada por par

Pedal, control de crucero, transmisión, ESP y protecciones expresan sus pedidos en par. El coordinador los arbitra y una cadena de mapas convierte el par final en combustible por cilindro y en un reparto de inyecciones.



Estructura de par del capítulo 12.

Adaptaciones

La ECU corrige el caudal de cada inyector durante su vida (pequeñas inyecciones de prueba en retención), regula el ralentí, amortigua los tirones del tren motriz y equilibra los cilindros midiendo la velocidad del cigüeñal.

Lazo de aire

El turbo de geometría variable (VGT) ajusta la presión de carga; la recirculación de gases (EGR) de alta y baja presión baja la temperatura de combustión y los NOx. El sensor lambda cierra el lazo.

Límite de humos

Si se inyecta más combustible del que el aire puede quemar aparece humo negro. Con la masa de aire medida, la ECU limita el combustible a $m_{aire}/(\lambda_{mín}\cdot 14{,}5)$.

Postratamiento

El catalizador de oxidación (DOC) quema CO e hidrocarburos; el filtro de partículas (DPF) retiene hollín y se regenera quemándolo; el SCR reduce los NOx con amoníaco obtenido del AdBlue. Euro 7 endurece límites, durabilidad y vigilancia a bordo.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Combustible por cilindro	$m_f = 4\pi \cdot T / (\eta \cdot H_u \cdot z)$	Motor de 4 tiempos, z cilindros.
Límite de humos	$m_{f,máx} = m_{aire} / (\lambda_{mín} \cdot 14{,}5)$	Diésel estequiométrico 14,5:1.
SCR	$4NO + 4NH_3 + O_2 \rightarrow 4N_2 + 6H_2O$	Reacción estándar.
Urea	$CO(NH_2)_2 + H_2O \rightarrow 2NH_3 + CO_2$	AdBlue: 32,5 % de urea.

Ejemplo resuelto: par pedido frente a límite de humos

1. Motor de 4 cilindros, η = 0,40, H_u = 42,8 MJ/kg; se piden 300 N·m.
2. $m_f = 4\pi \cdot 300/(0{,}40 \cdot 42{,}8 \cdot 10^6 \cdot 4) = 55{,}1$ mg por cilindro.
3. Con 780 mg de aire y $\lambda_{mín}$ = 1,2, el máximo es 44,8 mg.
4. Se inyectan 44,8 mg: el motor entrega 244 N·m hasta que el turbo aporte más aire.

Con 139 mg/s de NOx (como NO_2) el SCR necesita 0,279 g/s de AdBlue, unos 0,92 L/h (edc_par_humos_scr.py).

Postratamiento diésel

Componente	Elimina	Cómo
DOC	CO y HC	Oxidación catalítica
DPF	Partículas (hollín)	Filtrado y regeneración
SCR	NOx	Reducción con NH_3 del AdBlue
ASC	NH_3 sobrante	Catalizador de bloqueo de amoníaco

Código del capítulo

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

edc_par_humos_scr.py — conversión de par a combustible, límite de humos y consumo de AdBlue.

```
"""
edc_par_humos_scr.py — Capítulo 12: de par pedido a combustible, límite de humos por lambda
mínima y dosificación de AdBlue en el SCR.
"""
HU, ETA, AFR, CIL = 42.8e6, 0.40, 14.5, 4


def combustible_por_cilindro(par_nm):
    """Motor de 4 tiempos: trabajo por ciclo = 4π·T, repartido entre los cilindros (mg)."""
    return 4 * 3.14159265 * par_nm / (ETA * HU) / CIL * 1e6


def par_desde_combustible(mg_cil):
    return mg_cil * 1e-6 * CIL * ETA * HU / (4 * 3.14159265)
```

```
def limite_humos(aire_mg, lambda_min=1.2):
    return aire_mg / (lambda_min * AFR)


def adblue(nox_mg_s, anr=1.0):
    """NOx como NO2 equivalente (46 g/mol); 1 mol de urea da 2 de NH3; AdBlue = 32,5 % urea."""
    mol_nox = nox_mg_s / 1000 / 46.0
    urea_g_s = mol_nox * anr / 2 * 60.06
    return urea_g_s / 0.325


if __name__ == '__main__':
    pedido = 300.0
    mf = combustible_por_cilindro(pedido)
    lim = limite_humos(780.0)
    final = min(mf, lim)
    print(f'Par pedido {pedido:.0f} N·m -> {mf:.1f} mg/cilindro')
    print(f'Aire 780 mg/cilindro, lambda mín 1,2 -> máximo {lim:.1f} mg')
    print(f'Se inyectan {final:.1f} mg -> par entregado {par_desde_combustible(final):.0f} N·m')
    for nox in (50.0, 139.0):
        g = adblue(nox)
        print(f'NOx {nox:5.1f} mg/s -> AdBlue {g:.3f} g/s = {g * 3600 / 1090:.2f} L/h')
```

Errores frecuentes

- Anular el límite de humos para "ganar potencia": aparece hollín y el DPF se satura.
- Dosificar AdBlue con el SCR frío (no reacciona y se cristaliza).
- Interrumpir regeneraciones del DPF con trayectos cortos repetidos.

Autoevaluación

1. ¿Qué gobierna la estructura de la EDC?
2. ¿Qué reduce el SCR?
3. ¿Qué porcentaje de urea tiene el AdBlue?

Respuestas. 1. El par. **2.** Los NOx. **3.** 32,5 %.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Dirección eléctrica y by-wire, climatización y seguridad

Capítulo 13 · En el video: **minuto 2:58:14** · Duración 14:25

Objetivo. Describir los sistemas de dirección, confort y acceso y calcular la detección antipinzamiento y la distancia UWB.

Ideas clave

Dirección activa y EPS

Un engranaje planetario en la columna suma un ángulo del motor al del conductor: relación directa en ciudad e indirecta en carretera. La EPS ayuda con un motor eléctrico solo cuando hace falta (menos consumo que la hidráulica).

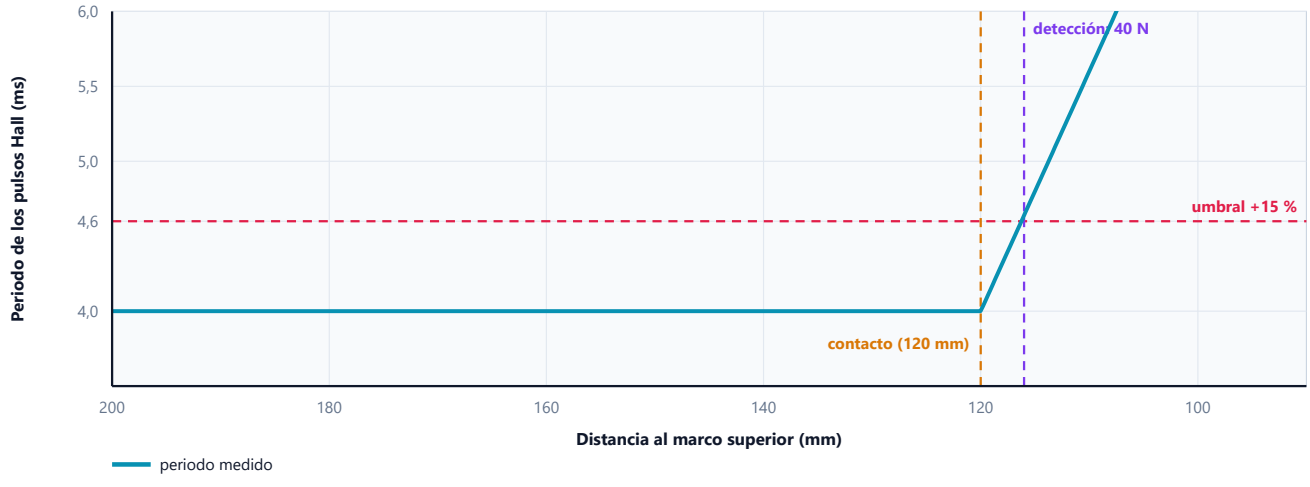
Steer-by-wire

Sin unión mecánica entre volante y ruedas: un actuador mueve la cremallera y otro da sensación al volante. Exige redundancia completa; algunos diseños conservan un respaldo mecánico con embrague. Permite estabilizar la guiñada dirigiendo además de frenando.

Actuadores de confort

Alzavidrios, techos corredizos, asientos con memoria y columnas eléctricas usan motores DC con sensores Hall. La protección antipinzamiento detecta que el motor se frena y lo invierte antes de que la fuerza supere 100 N.

Alzavidrios con protección antipinzamiento cursosmecatronica.com



Al tocar el obstáculo el motor se frena y los pulsos se espacian. Se invierte el giro a 116 mm, con 40 N: muy bajo el límite de 100 N.

Salida de antipinzamiento_y_uwb.py.

Climatización

El circuito frigorífico (compresor, condensador, expansión, evaporador) puede invertirse en una bomba de calor: en los eléctricos calienta con un tercio o la mitad de la energía de una resistencia. La climatización multizona balancea el calor del sol, las personas y la ventilación.

Acceso y seguridad

Inmovilizador con transpondedor, cierre "open-by-wire", llave digital en el teléfono y biometría. La banda ultraancha (UWB) mide la distancia por tiempo de vuelo y frustra los ataques de relé que engañaban a las llaves por intensidad de señal.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Relación variable	$\delta_{rueda} = \delta_{volante}/i + \delta_{motor}$	Superposición en la columna.
Fuerza de apriete	$F = k \cdot x$	Ensayo con rigidez de 10 N/mm.
UWB doble ida y vuelta	$t_v = (R_1R_2 - D_1D_2)/(R_1 + R_2 + D_1 + D_2)$	$d = c \cdot t_v$.
Bomba de calor	$COP = Q_{calor}/W_{eléctrica}$	2 a 3 en invierno.

Ejemplo resuelto: alzavidrios con un obstáculo

1. Obstáculo a 120 mm del marco; rigidez de ensayo 10 N/mm.
2. Periodo base de los pulsos Hall: 4,0 ms; umbral de detección +15 % (4,6 ms).
3. La fuerza llega a 37,5 N a 116,25 mm; la detección se produce a 116,0 mm con 40 N.
4. El motor invierte y abre la ventana.

Muy por debajo del límite de 100 N. En UWB, 2,40 m equivalen a 8,01 ns de vuelo: 1 ns de error son 30 cm (antipinzamiento_y_uwb.py).

Sistemas de dirección

Sistema	Unión mecánica	Asistencia	Función extra
Hidráulica	Sí	Bomba permanente	—
EPS	Sí	Motor eléctrico	Mantenimiento de carril, estacionamiento
Dirección activa	Sí	EPS + planetario	Relación variable
Steer-by-wire	No (o respaldo)	Actuador en la cremallera	Relación libre, conducción automatizada

Código del capítulo

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

antipinzamiento_y_uwb.py — detección antipinzamiento por periodo Hall, distancia UWB y consumo de una bomba de calor.

```
"""
antipinzamiento_y_uwb.py — Capítulo 13: protección antipinzamiento de un alzavidrios con
sensor Hall, distancia por UWB (doble ida y vuelta) y COP de una bomba de calor.
"""
C = 299_792_458.0


def periodos_hall(obstaculo_mm=120.0, k_n_mm=10.0, pulsos_mm=2.0):
    """Periodo (ms) de los pulsos Hall al subir; al tocar el obstáculo la fuerza crece
```

```
    con la rigidez k y el motor se frena."""
    datos, pos = [], 400.0               # mm desde el marco superior
    while pos > 0:
        fuerza = max(0.0, (obstaculo_mm - pos) * k_n_mm) if pos < obstaculo_mm else 0.0
        periodo = 4.0 * (1 + fuerza / 250.0)
        datos.append((pos, periodo, fuerza))
        pos -= 1 / pulsos_mm
    return datos


def detectar(datos, umbral=0.15):
    base = datos[0][1]
    for pos, periodo, fuerza in datos:
        if 4 <= pos <= 200 and periodo > base * (1 + umbral):
            return pos, fuerza
    return None


def distancia_ds_twr(t_round1, t_reply1, t_round2, t_reply2):
    tof = (t_round1 * t_round2 - t_reply1 * t_reply2) / (t_round1 + t_round2 + t_reply1 + t_reply2)
    return tof * C


if __name__ == '__main__':
    pos, f = detectar(periodos_hall())
    print(f'Obstáculo a 120 mm del marco: detectado a {pos:.1f} mm con {f:.1f} N -> el motor invierte')
    print('  (el reglamento exige no superar 100 N de fuerza de apriete)')

    d_real = 2.40
    tof = d_real / C
    r1, p1 = 2 * tof + 200e-6, 200e-6           # el llavero responde tras 200 µs
    r2, p2 = 2 * tof + 180e-6, 180e-6
    print(f'\nUWB: tiempo de vuelo {tof * 1e9:.2f} ns -> distancia {distancia_ds_twr(r1, p1, r2, p2):.2f} m')
    print(f'Un error de 1 ns en el tiempo de vuelo equivale a {C * 1e-9 * 100:.0f} cm')

    for t_ext, cop in ((5, 3.0), (-10, 2.0)):
        print(f'Bomba de calor a {t_ext:+d} °C, COP {cop}: 3 kW de calefacción consumen {3 / cop:.1f} kW'
              f' (resistencia PTC: 3,0 kW)')
```

Errores frecuentes

- Fijar el umbral antipinzamiento sin compensar el frío (los sellos frenan más el vidrio).
- Confiar en la intensidad de señal para saber dónde está la llave.
- Calentar un eléctrico solo con resistencias y perder autonomía.

Autoevaluación

1. ¿Qué fuerza máxima de apriete permite la protección antipinzamiento?
2. ¿Cuánto equivale 1 ns de tiempo de vuelo?
3. ¿Qué COP tiene una bomba de calor típica en invierno?

Respuestas. 1. 100 N. **2.** Unos 30 cm. **3.** Entre 2 y 3.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Compatibilidad electromagnética, plataformas de 800 V y diagnóstico a bordo

Capítulo 14 · En el video: minuto 3:12:39 · Duración 14:29

Objetivo. Identificar cómo se acoplan las perturbaciones, dimensionar un filtro y leer códigos de falla.

Ideas clave

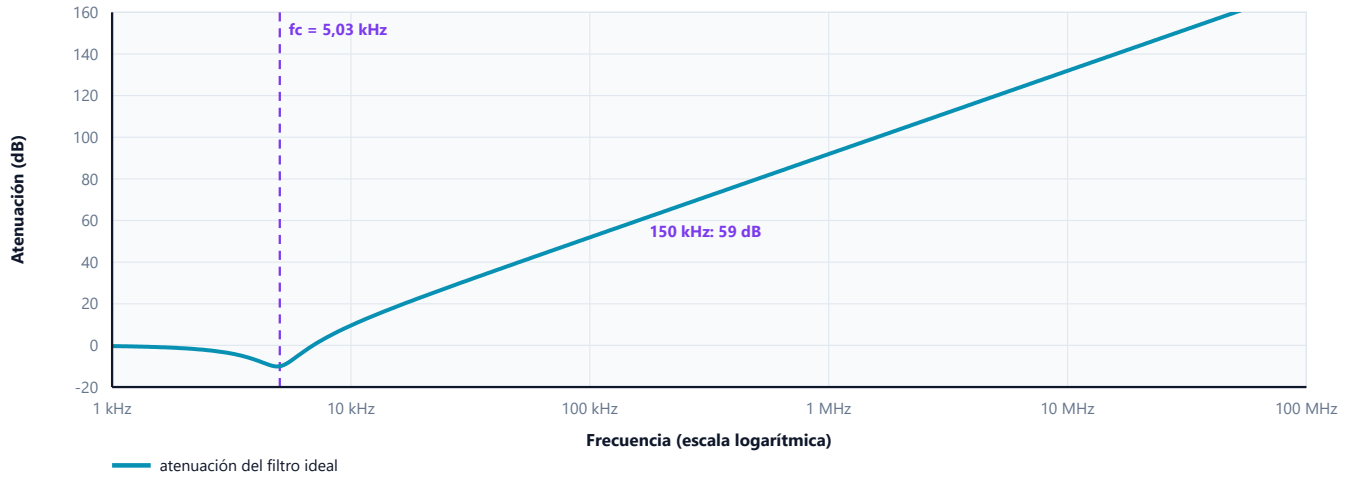
Fuente, acoplamiento y víctima

Toda perturbación tiene una fuente (inversor, encendido, motor), un camino y una víctima (sensor, radio). Los caminos son galvánico (masa compartida), capacitivo (campo eléctrico), inductivo (campo magnético) y radiado.

Transitorios y 800 V

La red de 12 V sufre descargas de carga y arranques en frío. Los inversores de carburo de silicio (SiC) de 400 y 800 V conmutan cientos de voltios en nanosegundos: sus flancos generan ruido de alta frecuencia que se controla con filtros LC, blindajes y buen trazado.

Filtro LC de entrada 10 µH / 100 µF (carga de 1 Ω) cursosmecatronica.com



Cerca de fc el filtro poco amortiguado resuena; sobre ella atenúa 40 dB por década. En la realidad, la inductancia parásita del condensador y la capacidad de la bobina limitan la atenuación sobre unos MHz.

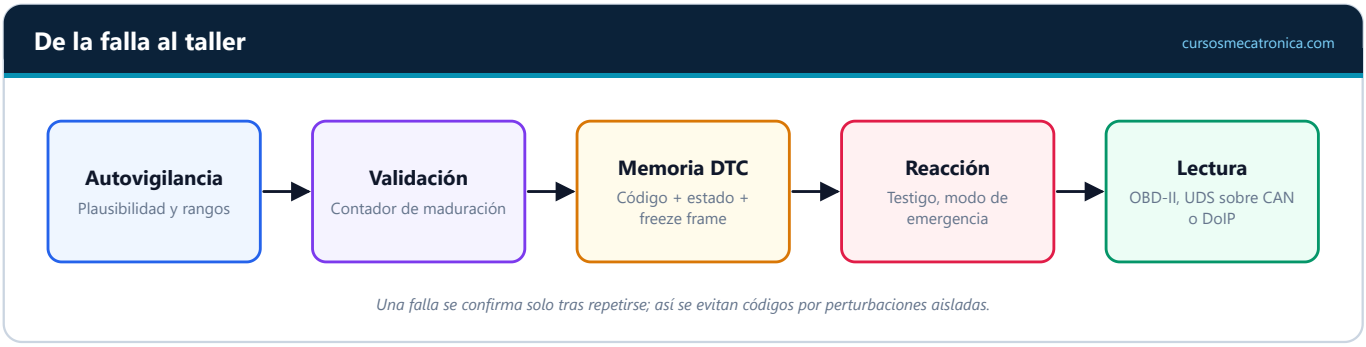
Salida de dtc_y_filtro_cem.py.

Ensayos

Cámara semianecoica para emisiones e inmunidad radiada, inyección de corriente en el cable (BCI), celda TEM; homologación con UNECE R10.

Autovigilancia

La ECU comprueba rangos, plausibilidad entre señales redundantes y sus propias etapas de potencia. Una falla se valida con un contador antes de registrarse con un código (DTC), un byte de estado y una "foto" de las condiciones (freeze frame). Si es grave, el vehículo pasa a modo de emergencia (limp-home).



Diagnóstico a bordo del capítulo 14.

OBD y UDS

OBD-II y EOBD vigilan todo lo que afecta a las emisiones y encienden el testigo; Euro 7 agrega vigilancia continua (OBM). En taller se usa UDS (ISO 14229) sobre CAN o sobre Ethernet (DoIP): lectura de códigos, valores, rutinas, programación de ECU y acceso con seguridad.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Frecuencia de corte LC	$f_c = 1 / (2\pi\sqrt{(LC)})$	10 µH y 100 µF: 5,03 kHz.
Atenuación ideal	$A \approx 40 \cdot \log_{10}(f/f_c)$ dB	40 dB por década.
Acoplamiento capacitivo	$i = C \cdot dv/dt$	Flancos rápidos de SiC.
DTC OBD	2 bytes: letra (P, C, B, U) + 4 dígitos	P0301 = falla de encendido cil. 1.

Ejemplo resuelto: filtro de entrada y lectura de códigos

1. L = 10 µH, C = 100 µF: $f_c = 1/(2\pi\sqrt{(10^{-9})}) = 5{,}03$ kHz.
2. A 150 kHz (inicio de la banda de radio de onda larga en ensayos) atenúa 59 dB; a 1 MHz, 92 dB (ideal).
3. Respuesta UDS 59 02 FF 03 01 00 2F: código P0301-00 con estado 0x2F.
4. Bits de 0x2F: falla ahora, falla en este ciclo, pendiente, confirmada y fallada desde el último borrado.

El filtro ideal no basta sobre unos MHz por los elementos parásitos. dtc_y_filtro_cem.py decodifica también un código de red U0123 pendiente.

Byte de estado de un DTC UDS

Bit	Nombre	Significado
0	testFailed	Falla presente ahora
1	testFailedThisOperationCycle	Falló en este ciclo
2	pendingDTC	Pendiente de confirmar
3	confirmedDTC	Confirmada y memorizada
4	testNotCompletedSinceLastClear	Sin probar desde el borrado
5	testFailedSinceLastClear	Falló desde el borrado
6	testNotCompletedThisOperationCycle	Sin probar en este ciclo
7	warningIndicatorRequested	Pide encender el testigo

Código del capítulo

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

dtc_y_filtro_cem.py — atenuación de un filtro LC y decodificación de códigos OBD y UDS con su estado.

```
"""
dtc_y_filtro_cem.py — Capítulo 14: atenuación de un filtro LC de entrada y decodificación
de códigos de falla OBD y UDS.
"""
import math


def atenuacion_db(f, l_h, c_f, r_carga=1.0):
    w = 2 * math.pi * f
    real = 1 - w * w * l_h * c_f
    imag = w * l_h / r_carga
    return -20 * math.log10(1 / math.hypot(real, imag))


def obd_dtc(byte_a, byte_b):
    letra = 'PCBU'[byte_a >> 6]
    return f'{letra}{(byte_a >> 4) & 3}{byte_a & 0xF:X}{byte_b >> 4:X}{byte_b & 0xF:X}'


ESTADO = ['testFailed', 'testFailedThisOperationCycle', 'pendingDTC', 'confirmedDTC',
          'testNotCompletedSinceLastClear', 'testFailedSinceLastClear',
          'testNotCompletedThisOperationCycle', 'warningIndicatorRequested']


def uds_19_02(respuesta):
    """Respuesta positiva a ReadDTCInformation (0x19 0x02): 59 02 máscara [DTC(3) estado(1)]..."""
    assert respuesta[0] == 0x59 and respuesta[1] == 0x02
    salida = []
    for k in range(3, len(respuesta), 4):
        hi, mid, low, st = respuesta[k:k + 4]
        codigo = obd_dtc(hi, mid) + f'-{low:02X}'
        bits = [ESTADO[b] for b in range(8) if st >> b & 1]
        salida.append((codigo, st, bits))
    return salida


if __name__ == '__main__':
    l, c = 10e-6, 100e-6
```

```
    fc = 1 / (2 * math.pi * math.sqrt(l * c))
    print(f'Filtro LC 10 µH / 100 µF: frecuencia de corte {fc / 1000:.2f} kHz')
    for f in (150e3, 1e6, 30e6):
        print(f'  {f / 1e6:6.2f} MHz: atenuación {atenuacion_db(f, l, c):5.1f} dB'
              f' (ideal −40 dB/década: {40 * math.log10(f / fc):5.1f} dB)')

    print(f'\nOBD modo 03, bytes 03 01 -> {obd_dtc(0x03, 0x01)}  (falla de encendido cilindro 1)')
    print(f'OBD modo 03, bytes 41 23 -> {obd_dtc(0x41, 0x23)}')
    resp = [0x59, 0x02, 0xFF, 0x03, 0x01, 0x00, 0x2F, 0xC1, 0x23, 0x87, 0x24]
    for codigo, st, bits in uds_19_02(resp):
        print(f'UDS {codigo} estado 0x{st:02X}: {", ".join(bits)}')
```

Errores frecuentes

- Borrar códigos sin anotar el freeze frame.
- Unir masas de potencia y de señal en cualquier punto (acoplamiento galvánico).
- Calcular el filtro con componentes ideales y olvidar sus parásitos.

Autoevaluación

1. ¿Cuáles son los cuatro caminos de acoplamiento?
2. ¿Qué norma define UDS?
3. ¿Qué indica el bit 3 del estado?

Respuestas. 1. Galvánico, capacitivo, inductivo y radiado. **2.** ISO 14229. **3.** Falla confirmada.

MANUAL DE ESTUDIO · MECATRÓNICA AUTOMOTRIZ

Resumen y cierre

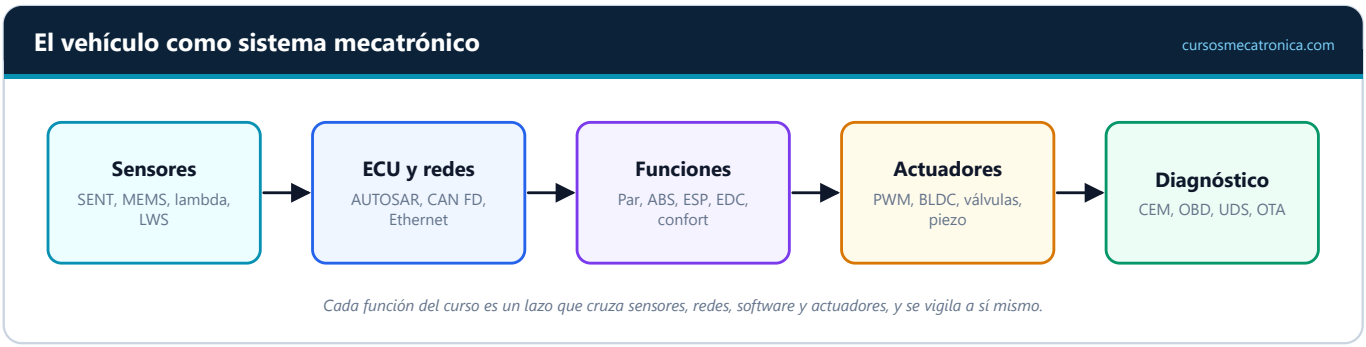
Cierre · En el video: **minuto 3:27:08** · Duración 0:53

Objetivo. Repasar cómo se conectan los sistemas del vehículo.

Ideas clave

Lo que se recorrió

Arquitectura y unidades de control, redes, sensores y actuadores; luego transmisión, frenos y estabilidad, inyección y gestión diésel, dirección, confort y acceso; y por último compatibilidad electromagnética y diagnóstico.



Síntesis del curso.

Para seguir

Con una interfaz OBD y un programa de diagnóstico, lee los valores en vivo de tu vehículo (régimen, lambda, presión del riel) y compáralos con los modelos de los programas del curso.

Fórmulas del curso

Tema	Relación
Carga de bus	$U = \Sigma(b/T)/\text{bitrate}$
Velocidad de rueda	$v = f/z \cdot \pi \cdot D$
Bobina	$L \cdot di/dt = d \cdot V - R \cdot i$
Deslizamiento	$\lambda = (v - \omega r)/v$
Guiñada de referencia	$\dot{\psi} = v\delta/[L(1 + Kv^2)]$
Tobera	$Q = C_d A\sqrt{(2\Delta p/\rho)}$
Filtro LC	$f_c = 1/(2\pi\sqrt{(LC)})$

Autoevaluación

1. ¿Qué comparten el ABS, el ESP y la EDC?
2. ¿Por qué todos los pedidos al motor se expresan en par?

Respuestas. 1. Son lazos cerrados que cruzan sensores, redes, software y actuadores. **2.** Para poder arbitrarlos y coordinarlos entre sistemas.