CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA

Video 06 — Parte 3B: Características Dinámicas, Sistemas de Orden Cero, Primer Orden (Prueba Experimental), Segundo Orden y Analogías — Orden Dinámico, Sistema de Orden Cero, Primer Orden y Ensayo Experimental, Respuesta al Escalón y en Frecuencia de Segundo Orden y Analogías

Parte del Curso: Parte 3B — Características Dinámicas: Sistemas de Orden Cero, Primer Orden (Prueba Experimental), Segundo Orden y Analogías
Arquitectura / Controlador Estudiado: Dinámica de Orden 0 (y = Kx), 1.er Orden (Constante τ y Ensayo Experimental Semilogarítmico), 2.do Orden (ω_n, ζ) y Analogías

$$1.er\ Orden:\ \tau \cdot dy/dt + y = K \cdot x,\ Z(t) = \ln(1 - y/y_{\infty}) = -t/\tau \mid 2.do\ Orden:\ (1/\omega_n^2)y'' + (2\zeta/\omega_n)y' + y = K \cdot x$$

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Características dinámicas y orden del sistema según el número de almacenadores independientes de energía; sistema de orden cero (y = K·x, sensibilidad estática K y respuesta instantánea sin desfase).
- Sistema de primer orden (τ = R·C o L/R): respuesta al escalón (63.2% en 1τ, 95% en 3τ, 99.3% en 5τ), caída de -20 dB/década y prueba experimental en laboratorio verificando linealidad de ln(1 - y/y_∞) = -t/τ.
- Sistema de segundo orden (circuito RLC serie con ω_n = 1/√(LC) y ζ = (R/2)√(C/L)): regímenes sobreamortiguado (ζ > 1), críticamente amortiguado (ζ = 1) y subamortiguado (0 < ζ < 1 con t_r, t_p, %OS y t_s).
- Respuesta en frecuencia de segundo orden (frecuencia de resonancia ω_r, pico M_r, factor de calidad Q = 1/(2ζ) y caída de -40 dB/década) y modelado multidominio mediante analogías eléctrico-mecánico-fluídico-térmicas.

2. Parámetros Nominales de Configuración del Controlador (Video 06)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	3.8	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.5 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.08 s, N = 12	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	1 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[0, 10]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[0, 100]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Características Dinámicas de los Sistemas y Orden del Modelo

Elementos Almacenadores de Energía Independientes, Ecuaciones Diferenciales Ordinarias, Régimen Transitorio y Estacionario

- **Turno 1 (Masculina):** Las características dinámicas de un sistema describen cómo evoluciona su salida en el tiempo cuando la entrada experimenta cambios rápidos o variaciones continuas.
- **Turno 2 (Femenina):** El orden dinámico de un circuito o instrumento es igual al orden de la derivada más alta en su ecuación diferencial, lo cual coincide con el número de elementos independientes que almacenan energía, como capacitores e inductores.
- **Turno 3 (Masculina):** Ante cualquier excitación, la respuesta total ye de te se compone de dos partes: la respuesta transitoria o natural, que depende de las raíces características del sistema y se extingue con el tiempo, y la respuesta estacionaria o forzada.
- **Turno 4 (Femenina):** De acuerdo con su complejidad dinámica, la gran mayoría de bloques electrónicos y sensores se modelan mediante sistemas de orden cero, de primer orden o de segundo orden.

Diapositiva 02: Sistema de Orden Cero: Respuesta Instantánea sin Memoria

Ecuación Algebraica y(t) = K·x(t), Sensibilidad Estática K, Desfase Nulo y Ejemplo del Divisor Resistivo / Potenciómetro

- **Turno 1 (Masculina):** Un sistema de orden cero es aquel que carece por completo de elementos almacenadores de energía, por lo que su relación entre entrada y salida no contiene derivadas ni integrales respecto al tiempo.
- **Turno 2 (Femenina):** Su comportamiento queda gobernado por la ecuación algebraica ye de te igual a ka por equis de te, donde la constante ka recibe el nombre de ganancia estática o sensibilidad estática del sistema.
- **Turno 3 (Masculina):** Como consecuencia directa, un sistema ideal de orden cero responde de manera instantánea ante cualquier escalón sin retardo temporal, con ancho de banda infinito y desfase exactamente igual a cero grados en toda frecuencia.
- **Turno 4 (Femenina):** El ejemplo clásico en electrónica es un divisor de tensión puramente resistivo o un potenciómetro sensor de posición a bajas frecuencias, antes de que las capacitancias parásitas de los cables se vuelvan significativas.

Diapositiva 03: Sistema de Primer Orden: Constante de Tiempo y Ecuación Diferencial

Modelo Estándar τ·(dy/dt) + y = K·x(t), Constante de Tiempo τ = R·C o τ = L/R y Función de Respuesta en Frecuencia

- **Turno 1 (Masculina):** Un sistema de primer orden contiene un único elemento almacenador de energía rodeado de elementos disipativos, como ocurre en un filtro pasivo erre ce, un circuito erre ele o un sensor térmico.
- **Turno 2 (Femenina):** Al aplicar las leyes de Kirchhoff a un circuito erre ce serie obtenemos la ecuación diferencial canónica de primer orden: tau por la derivada de la salida respecto al tiempo más la salida es igual a la ganancia estática ka por la entrada.
- **Turno 3 (Masculina):** El parámetro fundamental tau se denomina constante de tiempo del sistema, medida en segundos, y vale erre por ce en una red capacitiva o ele dividido entre erre en una red inductiva.
- **Turno 4 (Femenina):** Al sustituir la derivada por jota omega obtenemos su función de transferencia senoidal: ka dividido entre uno más jota omega tau, cuya frecuencia de esquina o de corte ocurre exactamente en omega sub ce igual a uno sobre tau.

Diapositiva 04: Respuestas Temporal y en Frecuencia de un Sistema de Primer Orden

Respuesta al Escalón y(t) = K·A·(1 - e^(-t/τ)), Regla del 63.2% en 1τ y 99.3% en 5τ, y Caída de -20 dB/década

- **Turno 1 (Masculina):** Cuando aplicamos un escalón de amplitud A a un sistema de primer orden inicialmente en reposo, la salida crece exponencialmente siguiendo la ley ka por A multiplicado por uno menos e elevado a menos te sobre tau.
- **Turno 2 (Femenina):** Al transcurrir exactamente una constante de tiempo tau, la salida alcanza el sesenta y tres punto dos por ciento de su cambio final; en tres tau alcanza el noventa y cinco por ciento y en cinco tau llega al noventa y nueve punto tres por ciento, considerándose establecido.
- **Turno 3 (Masculina):** Además, la pendiente inicial de la curva en el instante te igual a cero es máxima e intersecta la asíntota del valor final exactamente en el instante te igual a tau.
- **Turno 4 (Femenina):** En el dominio de la frecuencia, a la frecuencia de corte uno sobre tau la ganancia cae tres decibelios con un desfase de menos cuarenta y cinco grados, y para frecuencias altas la ganancia desciende a razón de menos veinte decibelios por década tendiendo la fase a menos noventa grados.

Diapositiva 05: Prueba Experimental de un Sistema de Primer Orden

Identificación en Laboratorio mediante Escalón, Función de Error Incompleto Z(t) = ln(1 - y/y_∞) y Verificación de Primer Orden

- **Turno 1 (Masculina):** En el laboratorio a menudo necesitamos determinar experimentalmente si un sensor o circuito se comporta realmente como un sistema de primer orden y medir con precisión su ganancia ka y su constante de tiempo tau.
- **Turno 2 (Femenina):** El ensayo experimental consiste en aplicar un cambio escalón conocido en la entrada, registrar la curva temporal ye de te con un osciloscopio o sistema de adquisición y medir primero el valor estacionario final ye infinito.
- **Turno 3 (Masculina):** Una primera estimación rápida obtiene tau leyendo en el eje horizontal el tiempo exacto en el que la respuesta cruza el sesenta y tres punto dos por ciento del salto total.
- **Turno 4 (Femenina):** Sin embargo, el método riguroso consiste en graficar el logaritmo natural de la fracción de respuesta incompleta frente al tiempo: si los datos experimentales forman una línea recta perfecta, se confirma que el sistema es de primer orden y la pendiente de esa recta es exactamente menos uno sobre tau.

Diapositiva 06: Sistema de Segundo Orden: Circuito RLC, Frecuencia Natural y Amortiguamiento

Ecuación Canónica (1/ω_n²)·y'' + (2ζ/ω_n)·y' + y = K·x(t), Frecuencia Natural ω_n = 1/√(LC) y Factor de Amortiguamiento ζ

- **Turno 1 (Masculina):** Un sistema de segundo orden posee dos elementos almacenadores de energía capaces de intercambiar energía entre sí, como la bobina y el capacitor en un circuito erre ele ce o la masa y el resorte en un acelerómetro.
- **Turno 2 (Femenina):** Su dinámica está gobernada por dos parámetros adimensionales y físicos esenciales: la frecuencia natural no amortiguada omega sub ene y la relación o factor de amortiguamiento zeta.
- **Turno 3 (Masculina):** En un circuito erre ele ce en serie, la frecuencia natural omega sub ene vale uno dividido entre la raíz cuadrada del producto de la inductancia ele por la capacitancia ce, expresada en radianes por segundo.
- **Turno 4 (Femenina):** Por su parte, el factor de amortiguamiento zeta vale la mitad de la resistencia erre multiplicada por la raíz cuadrada de ce sobre ele, cuantificando cuánta energía disipa el resistor en cada ciclo de oscilación.

Diapositiva 07: Respuesta al Escalón de un Sistema de Segundo Orden

Casos Sobreamortiguado ($\zeta > 1$), Críticamente Amortiguado ($\zeta = 1$) y Subamortiguado ($0 < \zeta < 1$ con Frecuencia $\omega_d = \omega_n \cdot \sqrt{(1 - \zeta^2)}$)

- **Turno 1 (Masculina):** El valor del factor de amortiguamiento zeta clasifica la respuesta al escalón de un sistema de segundo orden en tres regímenes dinámicos claramente diferenciados.
- **Turno 2 (Femenina):** Cuando zeta es mayor que uno, el sistema es sobreamortiguado: sus dos raíces reales negativas producen una respuesta lenta y monótona sin ninguna oscilación ni sobreimpulso.
- **Turno 3 (Masculina):** Cuando zeta es exactamente igual a uno, el sistema es críticamente amortiguado: logra la aproximación más rápida posible hacia el valor final sin llegar a sobrepasarlo jamás.
- **Turno 4 (Femenina):** Cuando zeta se encuentra entre cero y uno, el sistema es subamortiguado: presenta polos complejos conjugados y oscila alrededor del valor final a la frecuencia natural amortiguada omega sub de igual a omega sub ene por la raíz de uno menos zeta al cuadrado.

Diapositiva 08: Métricas de Desempeño Temporal en Sistemas de Segundo Orden

Tiempo de Subida t_r, Tiempo Pico $t_p = \pi/\omega_d$, Sobreimpulso Porcentual %OS y Tiempo de Establecimiento $t_s \approx 4/(\zeta \cdot \omega_n)$

- **Turno 1 (Masculina):** En un sistema de segundo orden subamortiguado caracterizamos la rapidez y la estabilidad transitoria mediante cuatro especificaciones temporales estándar.
- **Turno 2 (Femenina):** El tiempo de subida te sub erre mide lo que tarda la respuesta en ir del diez al noventa por ciento de su valor final, mientras que el tiempo pico te sub pe, igual a pi sobre omega sub de, es el instante en que alcanza el primer máximo.
- **Turno 3 (Masculina):** El sobreimpulso máximo porcentual depende exclusivamente del factor de amortiguamiento zeta según la exponencial de menos pi por zeta dividido entre la raíz de uno menos zeta al cuadrado.
- **Turno 4 (Femenina):** Finalmente, el tiempo de establecimiento al dos por ciento vale aproximadamente cuatro dividido entre el producto de zeta por omega sub ene; en instrumentación suele buscarse un amortiguamiento zeta entre cero punto seis y cero punto siete para equilibrar rapidez y bajo sobreimpulso.

Diapositiva 09: Respuesta en Frecuencia de un Sistema de Segundo Orden y Resonancia

Frecuencia de Resonancia $\omega_r = \omega_n \cdot \sqrt{(1 - 2\zeta^2)}$, Pico de Resonancia M_r, Factor de Calidad $Q = 1/(2\zeta)$ y Pendiente de -40 dB/década

- **Turno 1 (Masculina):** Al analizar la respuesta en frecuencia de un sistema de segundo orden, observamos que para frecuencias muy superiores a omega sub ene la magnitud cae con una pendiente pronunciada de menos cuarenta decibelios por década y la fase tiende a menos ciento ochenta grados.
- **Turno 2 (Femenina):** En la vecindad de la frecuencia natural, si el factor de amortiguamiento zeta es menor que uno sobre raíz de dos, es decir menor a cero punto setenta y siete, aparece una amplificación o pico de resonancia en la curva de magnitud.
- **Turno 3 (Masculina):** La frecuencia exacta donde ocurre ese máximo de ganancia es la frecuencia de resonancia omega sub erre, igual a omega sub ene multiplicada por la raíz de uno menos dos veces zeta al cuadrado.
- **Turno 4 (Femenina):** En circuitos sintonizados y filtros selectivos definimos el factor de calidad cu como uno sobre dos zeta: un factor cu elevado produce un pico de resonancia muy agudo y estrecho, ideal para seleccionar una portadora de radiofrecuencia.

Diapositiva 10: Modelado de Sistemas y Analogías Físicas Multidominio

Variables de Esfuerzo (Tensión, Fuerza, Presión, Temperatura) y Variables de Flujo (Corriente, Velocidad, Caudal, Flujo de Calor)

- **Turno 1 (Masculina):** Cerramos la Parte tres con un concepto unificador poderoso de la ingeniería: el modelado de sistemas mediante analogías físicas entre los dominios eléctrico, mecánico, hidráulico y térmico.
- **Turno 2 (Femenina):** Sistemas físicamente muy distintos obedecen ecuaciones diferenciales idénticas porque todos poseen variables de esfuerzo como voltaje, fuerza o presión, y variables de flujo como corriente, velocidad o caudal.
- **Turno 3 (Masculina):** En la analogía directa fuerza-voltaje, la masa mecánica que se opone a cambios de velocidad equivale a la inductancia ele, el amortiguador viscoso equivale a la resistencia erre y la flexibilidad del resorte equivale a la capacitancia ce.
- **Turno 4 (Femenina):** Gracias a estas analogías podemos simular cualquier mecanismo, suspensión, circuito neumático o disipador térmico como un circuito eléctrico equivalente y analizar su respuesta con las herramientas de esta Parte tres.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_06.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 06: Video 06 — Parte 3B: Características Dinámicas, Sistemas de Orden Cero, Primer Orden (Prueba
Experimental), Segundo Orden y Analogías
 * Archivo: controlador_tiempo_real_video_06.ino
 * ----------------------------------------------------------------------------
 * Descripción Técnica:
 *   Firmware determinista en C/C++ Embebido para Microcontrolador (compatible
 *   con arquitectura AVR / ARM Cortex-M STM32 / ESP32 / RP2040) que implementa:
 *     1. Acceso a nivel de registros con punteros 'volatile' y máscaras de bits.
 *     2. Controlador PID Discreto por Transformación Bilineal de Tustin con
 *        derivada sobre la variable medida (-PV), filtro derivativo N y
 *        esquema Anti-Windup por retro-cálculo y Clamping.
 *     3. Máquina de Estados Finitos (FSM de Moore) no bloqueante para
 *        secuenciación segura e interbloqueo de emergencia.
 *     4. Trama de telemetría con verificación CRC-16-CCITT hacia SBC/PLC.
 *
 * Enfoque del Módulo:
 *   Dinámica de Orden 0 (y = Kx), 1.er Orden (Constante τ y Ensayo Experimental Semilogarítmico), 2.do Orden
(ω_n, ζ) y Analogías
 * ============================================================================
 */

#include <Arduino.h>
#include <stdint.h>

// --- Definición de Pines de Interfase (Entradas Optoaisladas y Salidas PWM/MOSFET) ---
const uint8_t PIN_ADC_PV_SENSOR   = A0; // Entrada analógica PV (ej. Shunt 165 ohm de lazo 4-20 mA)
const uint8_t PIN_ADC_SP_CONSIGNA = A1; // Consigna local o referencia analógica (0..5V / 0..3.3V)
const uint8_t PIN_DI_MARCHA_NO    = 2;  // Entrada optoacoplada PC817: Pulsador de Marcha (NO)
const uint8_t PIN_DI_PARO_NC      = 3;  // Entrada optoacoplada PC817: Paro / Emergencia (Cableado NC)
const uint8_t PIN_PWM_ACTUADOR    = 9;  // Salida PWM hacia Driver MOSFET / Puente H
const uint8_t PIN_DO_SSR_ALARMA   = 8;  // Salida hacia Relé de Estado Sólido (SSR) / Piloto

// --- Parámetros Nominales del Controlador PID Discreto (Video 06) ---
const float KP_GAIN      = 3.800f;
const float TI_SEC       = 0.500f;
const float TD_SEC       = 0.080f;
const float N_FILT       = 12.0f;
const float TS_SEC       = 0.0010f; // Periodo de muestreo Ts = 1 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 10.0f;
const float PV_MIN_ENG   = 0.0f;
const float PV_MAX_ENG   = 100.0f;

// --- Variables Volátiles Compartidas con Interrupción (ISR) ---
volatile bool     g_flagTickControl = false;
volatile uint32_t g_contadorPulsosEncoder = 0;

// --- Estructura de Estado del Controlador PID Discreto (Tustin + Anti-Windup) ---
struct ControladorPIDDiscreto {
  float kp;
  float ki;
  float kd;
  float n_divisor;
  float ts;
  float u_min;
  float u_max;
  float integral_acum;
  float termino_d_prev;
  float pv_prev;
  float error_prev;
  float u_salida;
};

ControladorPIDDiscreto g_pid;
```

```
// --- Máquina de Estados Finitos (FSM No Bloqueante de Moore) ---
enum EstadoControlador : uint8_t {
  ESTADO_REPOSO_SEGURO   = 0,
  ESTADO_PRECARGA_SUAVE  = 1,
  ESTADO_LAZO_PID_ACTIVO = 2,
  ESTADO_FALLA_PARO_NC   = 3
};

EstadoControlador g_estadoActual = ESTADO_REPOSO_SEGURO;
uint32_t g_prevMicrosLazo  = 0;
uint32_t g_timestampEstado = 0;
uint32_t g_pasoMuestreo    = 0;

// --- Rutina de Servicio de Interrupción Externa (Encoder / Sensor Rápido) ---
void isrPulsoEncoder() {
  g_contadorPulsosEncoder++;
}

// --- Inicialización de Coeficientes del PID Discreto ---
void inicializarPID(ControladorPIDDiscreto &c) {
  c.kp             = KP_GAIN;
  c.ki             = (TI_SEC > 1e-5f) ? (KP_GAIN / TI_SEC) : 0.0f;
  c.kd             = KP_GAIN * TD_SEC;
  c.n_divisor      = N_FILT;
  c.ts             = TS_SEC;
  c.u_min          = U_MIN_LIM;
  c.u_max          = U_MAX_LIM;
  c.integral_acum  = 0.0f;
  c.termino_d_prev = 0.0f;
  c.pv_prev        = 0.0f;
  c.error_prev     = 0.0f;
  c.u_salida       = 0.0f;
}

// --- Evaluación en Tiempo Real del PID por Regla Trapezoidal de Tustin ---
float evaluarPasoPID_Tustin(ControladorPIDDiscreto &c, float sp, float pv) {
  float error = sp - pv;

  // 1. Término Proporcional
  float p_term = c.kp * error;

  // 2. Término Derivativo sobre -PV filtrado por Tustin (Evita Derivative Kick)
  float tau_d  = (c.kp > 1e-6f) ? ((c.kd / c.kp) / c.n_divisor) : 0.0f;
  float denom  = 2.0f * tau_d + c.ts;
  float alpha1 = (2.0f * tau_d - c.ts) / denom;
  float beta0  = (2.0f * c.kd) / denom;
  float d_term = alpha1 * c.termino_d_prev - beta0 * (pv - c.pv_prev);

  // 3. Candidato de Integración Trapezoidal (Tustin)
  float delta_i = 0.5f * c.ki * c.ts * (error + c.error_prev);
  float i_cand  = c.integral_acum + delta_i;

  // 4. Suma Pre-Saturación y Esquema Anti-Windup Condicional (Clamping)
  float u_unsat = p_term + i_cand + d_term;
  float u_sat   = u_unsat;
  if (u_unsat > c.u_max) {
    u_sat = c.u_max;
    if (error < 0.0f) c.integral_acum = i_cand; // Solo integra si ayuda a desaturar
  } else if (u_unsat < c.u_min) {
    u_sat = c.u_min;
    if (error > 0.0f) c.integral_acum = i_cand;
  } else {
    c.integral_acum = i_cand;
  }

  // Actualizar memoria de estados discretos z^-1
  c.termino_d_prev = d_term;
```

```
  c.pv_prev        = pv;
  c.error_prev     = error;
  c.u_salida       = u_sat;
  return u_sat;
}

// --- Escalado Lineal Equivalente a NORM_X + SCALE_X ---
inline float escalarCuentasADC(int cuentasCrudas, int rawMin, int rawMax, float engMin, float engMax) {
  float u_norm = (float)(cuentasCrudas - rawMin) / (float)(rawMax - rawMin);
  if (u_norm < 0.0f) u_norm = 0.0f;
  if (u_norm > 1.0f) u_norm = 1.0f;
  return engMin + u_norm * (engMax - engMin);
}

// --- Cálculo de Verificación de Redundancia Cíclica CRC-16-CCITT (Polinomio 0x1021) ---
uint16_t calcularCRC16(const uint8_t *datos, uint8_t longitud) {
  uint16_t crc = 0xFFFFU;
  for (uint8_t i = 0; i < longitud; i++) {
    crc ^= ((uint16_t)datos[i] << 8);
    for (uint8_t b = 0; b < 8; b++) {
      if (crc & 0x8000U) {
        crc = (crc << 1) ^ 0x1021U;
      } else {
        crc <<= 1;
      }
    }
  }
  return crc;
}

void setup() {
  Serial.begin(115200);
  pinMode(PIN_DI_MARCHA_NO, INPUT_PULLUP);
  pinMode(PIN_DI_PARO_NC, INPUT_PULLUP);
  pinMode(PIN_PWM_ACTUADOR, OUTPUT);
  pinMode(PIN_DO_SSR_ALARMA, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(PIN_DI_MARCHA_NO), isrPulsoEncoder, RISING);
  inicializarPID(g_pid);
  g_prevMicrosLazo  = micros();
  g_timestampEstado = millis();

  Serial.println(F("# ================================================================"));
  Serial.println(F("# CURSO CONTROLADORES EN MECATRONICA - VIDEO 06"));
  Serial.println(F("# Columnas: t_s,estado_fsm,SP_eng,PV_eng,u_PID,pulsos_enc,crc16"));
  Serial.println(F("# ================================================================"));
}

void loop() {
  uint32_t ahoraMicros = micros();
  uint32_t periodoMicros = (uint32_t)(TS_SEC * 1000000.0f);

  if (ahoraMicros - g_prevMicrosLazo >= periodoMicros) {
    g_prevMicrosLazo += periodoMicros;
    float t_sec = g_pasoMuestreo * TS_SEC;

    // 1. Lectura de Entradas Digitales (Seguridad de Paro NC y Marcha NO)
    bool paroCableadoOK = (digitalRead(PIN_DI_PARO_NC) == HIGH);
    bool ordenMarcha    = (digitalRead(PIN_DI_MARCHA_NO) == LOW);

    // 2. Lectura y Escalado de Entradas Analógicas (NORM_X -> SCALE_X)
    int rawPV = analogRead(PIN_ADC_PV_SENSOR);
    int rawSP = analogRead(PIN_ADC_SP_CONSIGNA);
    float pv_eng = escalarCuentasADC(rawPV, 0, 1023, PV_MIN_ENG, PV_MAX_ENG);
    float sp_eng = escalarCuentasADC(rawSP, 0, 1023, PV_MIN_ENG, PV_MAX_ENG);

    // 3. Evaluación de la Máquina de Estados Finitos (FSM de Moore No Bloqueante)
    if (!paroCableadoOK) {
```

```
    g_estadoActual = ESTADO_FALLA_PARO_NC;
  }

  float mandoActuador = 0.0f;
  switch (g_estadoActual) {
    case ESTADO_REPOSO_SEGURO:
      mandoActuador = 0.0f;
      digitalWrite(PIN_DO_SSR_ALARMA, LOW);
      if (ordenMarcha && paroCableadoOK) {
        g_timestampEstado = millis();
        g_estadoActual    = ESTADO_PRECARGA_SUAVE;
      }
      break;

    case ESTADO_PRECARGA_SUAVE:
      mandoActuador = 0.25f * U_MAX_LIM;
      if (millis() - g_timestampEstado >= 200UL) {
        // Transferencia sin saltos (Bumpless) al activar el lazo PID
        g_pid.integral_acum = mandoActuador;
        g_pid.pv_prev       = pv_eng;
        g_estadoActual      = ESTADO_LAZO_PID_ACTIVO;
      }
      break;

    case ESTADO_LAZO_PID_ACTIVO:
      mandoActuador = evaluarPasoPID_Tustin(g_pid, sp_eng, pv_eng);
      digitalWrite(PIN_DO_SSR_ALARMA, LOW);
      break;

    case ESTADO_FALLA_PARO_NC:
    default:
      mandoActuador = 0.0f;
      g_pid.integral_acum = 0.0f;
      digitalWrite(PIN_DO_SSR_ALARMA, HIGH);
      if (paroCableadoOK && ordenMarcha) {
        g_estadoActual = ESTADO_REPOSO_SEGURO;
      }
      break;
  }

  // 4. Escritura de Salida PWM hacia Etapa de Potencia (MOSFET / Puente H)
  float u_norm_pwm = (mandoActuador - U_MIN_LIM) / (U_MAX_LIM - U_MIN_LIM);
  int duty8bit     = constrain((int)(u_norm_pwm * 255.0f), 0, 255);
  analogWrite(PIN_PWM_ACTUADOR, duty8bit);

  // 5. Lectura Atómica de Contador de Interrupción y Telemetría con CRC-16
  if (g_pasoMuestreo % 10 == 0) {
    noInterrupts();
    uint32_t pulsosCopia = g_contadorPulsosEncoder;
    interrupts();

    uint8_t tramaBytes[6];
    tramaBytes[0] = (uint8_t)g_estadoActual;
    tramaBytes[1] = (uint8_t)duty8bit;
    tramaBytes[2] = (uint8_t)((rawPV >> 8) & 0xFF);
    tramaBytes[3] = (uint8_t)(rawPV & 0xFF);
    tramaBytes[4] = (uint8_t)((pulsosCopia >> 8) & 0xFF);
    tramaBytes[5] = (uint8_t)(pulsosCopia & 0xFF);
    uint16_t crc  = calcularCRC16(tramaBytes, 6);

    Serial.print(t_sec, 3);              Serial.print(',');
    Serial.print((int)g_estadoActual);   Serial.print(',');
    Serial.print(sp_eng, 2);             Serial.print(',');
    Serial.print(pv_eng, 2);             Serial.print(',');
    Serial.print(mandoActuador, 2);      Serial.print(',');
    Serial.print(pulsosCopia);           Serial.print(",0x");
    Serial.println(crc, HEX);
  }
```

```
    g_pasoMuestreo++;
  }
}
```

5. Script Python SBC: Sintonización, Índices IAE/ITAE y Tramas (supervision_sbc_y_sintonizacion_video_06.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
VIDEO 06: Video 06 — Parte 3B: Características Dinámicas, Sistemas de Orden Cero, Primer Orden (Prueba
Experimental), Segundo Orden y Analogías
Archivo: supervision_sbc_y_sintonizacion_video_06.py
--------------------------------------------------------------------------------
Descripción Técnica:
  Script integral en Python para Computadora de Placa Única (Raspberry Pi /
  SBC) y estación de ingeniería que implementa:
    1. Simulación en lazo cerrado de un controlador PID discreto (Tustin +
       Filtro Derivativo N + Anti-Windup por Clamping y Back-Calculation).
    2. Cálculo cuantitativo de métricas transitorias (tr, tp, Mp, ts) y de
       los cuatro índices integrales de desempeño (IAE, ISE, ITAE, ITSE).
    3. Reglas automáticas de sintonización de Ziegler-Nichols (1.º y 2.º
       método), Cohen-Coon y sintonización robusta IMC / Lambda.
    4. Construcción y validación de tramas binarias Maestro-Coprocesador
       (SBC <-> MCU/PLC) protegidas con suma de verificación CRC-16-CCITT.
================================================================================
"""

import math
import struct
from typing import Dict, List, Tuple

# Parámetros nominales del Video 06
KP_NOMINAL = 3.8
TI_NOMINAL = 0.5
TD_NOMINAL = 0.08
N_FILTRO   = 12
TS_SEG     = 0.0010
U_MIN      = 0
U_MAX      = 10


def sintonizar_pid_ziegler_nichols_y_lambda(
    K_planta: float, L_retardo: float, T_constante: float, lambda_deseado: float
) -> Dict[str, Dict[str, float]]:
    """
    Calcula las ganancias PID (Kp, Ti, Td) mediante:
      - Ziegler-Nichols 1.er Método (Curva de Reacción FOPDT)
      - Método de Cohen-Coon (para retardo significativo L/T)
      - Método IMC / Regla Lambda (respuesta sin sobreimpulso)
    """
    r = L_retardo / max(T_constante, 1e-6)

    # 1. Ziegler-Nichols Lazo Abierto (PID Estándar ISA)
    kp_zn = (1.2 * T_constante) / (K_planta * max(L_retardo, 1e-6))
    ti_zn = 2.0 * L_retardo
    td_zn = 0.5 * L_retardo

    # 2. Cohen-Coon (PID)
    kp_cc = (1.0 / (K_planta * r)) * (4.0 / 3.0 + r / 4.0)
    ti_cc = L_retardo * (32.0 + 6.0 * r) / (13.0 + 8.0 * r)
    td_cc = L_retardo * 4.0 / (11.0 + 2.0 * r)

    # 3. IMC / Lambda (PI robusto sin sobreimpulso + término D para retardo Padé)
    kp_imc = T_constante / (K_planta * (lambda_deseado + L_retardo))
    ti_imc = T_constante
    td_imc = (T_constante * L_retardo) / (2.0 * T_constante + L_retardo)

    return {
        "Ziegler_Nichols_FOPDT": {
            "Kp": round(kp_zn, 4), "Ti_s": round(ti_zn, 4), "Td_s": round(td_zn, 4)
        },
```

```
        "Cohen_Coon": {
            "Kp": round(kp_cc, 4), "Ti_s": round(ti_cc, 4), "Td_s": round(td_cc, 4)
        },
        "IMC_Lambda_Robusto": {
            "Kp": round(kp_imc, 4), "Ti_s": round(ti_imc, 4), "Td_s": round(td_imc, 4)
        },
    }


def simular_lazo_pid_discreto_tustin(
    kp: float, ti: float, td: float, n_filt: float, ts: float, pasos: int = 800
) -> Dict[str, float]:
    """
    Simula una planta mecatrónica de 2.º orden controlada por un PID discreto
    discretizado por Tustin con Anti-Windup y calcula tr, Mp, ts, IAE, ISE, ITAE.
    """
    ki = kp / ti if ti > 1e-6 else 0.0
    kd = kp * td
    tau_d = (td / n_filt) if n_filt > 0 else 0.0
    denom_d = 2.0 * tau_d + ts
    alpha_d = (2.0 * tau_d - ts) / denom_d
    beta_d  = (2.0 * kd) / denom_d

    # Estado de la planta (wn = 8 rad/s, zeta = 0.45, K = 1.0)
    wn = 8.0
    zeta = 0.45
    y_pos = 0.0
    y_vel = 0.0
    sp = 1.0

    integral = 0.0
    d_prev = 0.0
    y_prev = 0.0
    e_prev = 0.0

    iae = 0.0
    ise = 0.0
    itae = 0.0
    itse = 0.0
    y_max = 0.0
    t_pico = 0.0
    t_10 = None
    t_90 = None
    t_estable = 0.0

    for k in range(pasos):
        t = k * ts
        error = sp - y_pos

        # Acumulación de índices integrales de desempeño
        abs_e = abs(error)
        iae  += abs_e * ts
        ise  += (error ** 2) * ts
        itae += (t * abs_e) * ts
        itse += (t * (error ** 2)) * ts

        # Algoritmo PID Discreto con filtro derivativo sobre -PV
        p_term = kp * error
        d_term = alpha_d * d_prev - beta_d * (y_pos - y_prev)
        i_cand = integral + 0.5 * ki * ts * (error + e_prev)

        u_unsat = p_term + i_cand + d_term
        u_cmd = max(U_MIN, min(U_MAX, u_unsat))

        # Anti-Windup condicional (Clamping)
        if u_unsat == u_cmd or (u_unsat > U_MAX and error < 0) or (u_unsat < U_MIN and error > 0):
            integral = i_cand
```

```
        d_prev = d_term
        y_prev = y_pos
        e_prev = error

        # Integración de la planta física (Euler semi-implícito)
        acel = (wn ** 2) * u_cmd - 2.0 * zeta * wn * y_vel - (wn ** 2) * y_pos
        y_vel += acel * ts
        y_pos += y_vel * ts

        if y_pos >= 0.10 * sp and t_10 is None:
            t_10 = t
        if y_pos >= 0.90 * sp and t_90 is None:
            t_90 = t
        if y_pos > y_max:
            y_max = y_pos
            t_pico = t
        if abs(sp - y_pos) > 0.02 * sp:
            t_estable = t

    tr = (t_90 - t_10) if (t_10 is not None and t_90 is not None) else 0.0
    mp_pct = max(0.0, (y_max - sp) / sp * 100.0)

    return {
        "tiempo_subida_tr_s": round(tr, 4),
        "tiempo_pico_tp_s": round(t_pico, 4),
        "sobreimpulso_Mp_pct": round(mp_pct, 2),
        "tiempo_establecimiento_ts_2pct_s": round(t_estable, 4),
        "IAE": round(iae, 4),
        "ISE": round(ise, 4),
        "ITAE": round(itae, 4),
        "ITSE": round(itse, 4),
    }


def crc16_ccitt(data: bytes) -> int:
    """Calcula CRC-16-CCITT (0x1021, init 0xFFFF) para enlace serie SBC <-> MCU."""
    crc = 0xFFFF
    for byte in data:
        crc ^= (byte << 8)
        for _ in range(8):
            if crc & 0x8000:
                crc = ((crc << 1) ^ 0x1021) & 0xFFFF
            else:
                crc = (crc << 1) & 0xFFFF
    return crc


def empaquetar_trama_consigna_sbc(comando: int, setpoint_float: float) -> bytes:
    """Construye trama binaria: [0xAA, 0x55, CMD, LEN=4, float32_LE, CRC16_BE]."""
    cabecera_y_payload = struct.pack("<BBBf", 0xAA, 0x55, comando, 4, setpoint_float)
    crc = crc16_ccitt(cabecera_y_payload)
    return cabecera_y_payload + struct.pack(">H", crc)


if __name__ == "__main__":
    print("=" * 78)
    print(" CURSO: FUNDAMENTOS DE ELECTRÓNICA — VIDEO 06")
    print(" Enfoque:", "Dinámica de Orden 0 (y = Kx), 1.er Orden (Constante τ y Ensayo Experimental
Semilogarítmico), 2.do Orden (ω_n, ζ) y Analogías")
    print("=" * 78)

    tablas = sintonizar_pid_ziegler_nichols_y_lambda(
        K_planta=1.25, L_retardo=0.15, T_constante=0.85, lambda_deseado=0.35
    )
    print("\n[1] Sintonización Comparativa (Ziegler-Nichols, Cohen-Coon e IMC Lambda):")
    for metodo, gan in tablas.items():
        print(f"    {metodo:24s} -> Kp = {gan['Kp']:6.3f} | Ti = {gan['Ti_s']:6.3f} s | Td = {gan['Td_s']:6.3f}
s")
```

```
metricas = simular_lazo_pid_discreto_tustin(KP_NOMINAL, TI_NOMINAL, TD_NOMINAL, N_FILTRO, 0.005)
print("\n[2] Desempeño Temporal e Índices Integrales (IAE, ISE, ITAE, ITSE):")
for k_m, v_m in metricas.items():
    print(f"    {k_m:34s} : {v_m}")

trama = empaquetar_trama_consigna_sbc(comando=0x01, setpoint_float=50)
print("\n[3] Trama Binaria Híbrida SBC -> MCU con CRC-16-CCITT:", trama.hex(" ").upper())
```

6. Programa PLC IEC 61131-3 (ST + IL): Enclavamiento, GRAFCET y Escalado 4-20 mA (automatizacion_plc_iec61131_video_06.st)

```
(* ==========================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 06: Video 06 — Parte 3B: Características Dinámicas, Sistemas de Orden Cero, Primer Orden (Prueba
Experimental), Segundo Orden y Analogías
 * Archivo: automatizacion_plc_iec61131_video_06.st
 * --------------------------------------------------------------------------
 * Estándar: IEC 61131-3 — Texto Estructurado (ST) + Equivalencia en Lista de
 *           Instrucciones (IL) y Peldaños de Escalera (Ladder Diagram)
 * Descripción Técnica:
 *   Bloque de Función Industrial (FB_ControladorCeldaMecatronica_V06) que
 *   integra el Ciclo de Barrido (Scan), circuito de enclavamiento prioritario
 *   al paro (cableado NC), secuenciador GRAFCET de 4 etapas, temporizador TON,
 *   contador de lotes CTU, registro de corrimiento de rechazo de calidad SHRB,
 *   escalado analógico 4-20 mA (NORM_X / SCALE_X con detección de rotura de
 *   hilo < 3.6 mA) y algoritmo PID discreto con Anti-Windup.
 * ========================================================================== *)

FUNCTION_BLOCK FB_ControladorCeldaMecatronica_V06
VAR_INPUT
    (* Entradas Digitales de la Tabla de Imagen de Proceso PII (%IX) *)
    xPulsadorMarcha_NO   : BOOL; (* %IX0.0 - Contacto Normalmente Abierto *)
    xPulsadorParo_NC     : BOOL; (* %IX0.1 - Cableado Físicamente NC en Campo *)
    xReleTermico_NC      : BOOL; (* %IX0.2 - Guardamotor NC (1 = Sin Falla) *)
    xSensorPiezaEntrada  : BOOL; (* %IX0.3 - Sensor Inductivo PNP 24V DC *)
    xCamaraPiezaDefecto  : BOOL; (* %IX0.4 - 1 = Envase Defectuoso Detectado *)
    xPulsoEncoderBanda   : BOOL; (* %IX0.5 - Reloj de Desplazamiento de Banda *)
    xResetContadores     : BOOL; (* %IX0.6 - Reinicio Manual de Lote *)

    (* Entradas Analógicas Crudas del Convertidor ADC (%IW) *)
    iEntradaAnalogicaRaw : INT;  (* %IW64  - Lazo 4-20 mA: 0 a 27648 cuentas *)
    rSetpointIngenieria  : REAL := 60.0;
END_VAR

VAR_OUTPUT
    (* Salidas Digitales hacia la Tabla de Imagen de Salidas PIQ (%QX) *)
    xMotorBandaContactor : BOOL; (* %QX0.0 - Contactor Principal Enclavado *)
    xValvulaDosificadora : BOOL; (* %QX0.1 - Solenoide 24V DC con Diodo Flyback *)
    xCilindroExpulsor    : BOOL; (* %QX0.2 - Rechazo en Posición Bit 5 del SHRB *)
    xAlarmaRoturaCable   : BOOL; (* %QX0.3 - Corriente < 3.6 mA o Paro Activo *)
    xLoteCompletado      : BOOL; (* %QX0.4 - Salida Q del Contador CTU *)

    (* Salida Analógica Escalada y Variables de Diagnóstico HMI *)
    iSalidaAnalogicaRaw  : INT;  (* %QW80  - Mando 0..27648 hacia Variador/SSR *)
    rVariableProcesoPV   : REAL; (* Variable física escalada en unidades de ing. *)
    rSalidaControlPID    : REAL; (* Esfuerzo de control 0.0 a 100.0 % *)
    iEstadoGrafcet       : INT;  (* Etapa activa del GRAFCET: 0, 10, 20, 30 *)
END_VAR

VAR
    (* Instancias de Temporizadores y Contadores Estándar IEC 61131-3 *)
    tonDosificado        : TON;
    ctuContadorLotes     : CTU;

    (* Variables Internas de Escalado NORM_X / SCALE_X *)
    rValorNormalizado    : REAL;
    rMinIngenieria       : REAL := 0.0;
    rMaxIngenieria       : REAL := 100.0;

    (* Registro de Corrimiento de 16 Bits (Shift Register para Rechazo) *)
    wRegistroCorrimiento : WORD := 16#0000;
    xPulsoEncoderPrev    : BOOL := FALSE;
    xFlancoEncoder       : BOOL := FALSE;

    (* Memoria Retentiva del Bloque PID Discreto *)
    rKp                  : REAL := 3.800;
    rTi_s                : REAL := 0.500;
    rTd_s                : REAL := 0.080;
```

```
    rTs_s                : REAL := 0.0010;
    rIntegralAcum        : REAL := 0.0;
    rErrorPrev           : REAL := 0.0;
    rPvPrev              : REAL := 0.0;
END_VAR

(* ================================================================================
 * 1. EQUIVALENCIA EN LISTA DE INSTRUCCIONES (IL) DEL PELDAÑO DE ENCLAVAMIENTO:
 *      LD     xPulsadorMarcha_NO
 *      OR     xMotorBandaContactor
 *      AND    xPulsadorParo_NC
 *      AND    xReleTermico_NC
 *      ANDN   xAlarmaRoturaCable
 *      ST     xMotorBandaContactor
 * ================================================================================ *)
xMotorBandaContactor := (xPulsadorMarcha_NO OR xMotorBandaContactor)
                        AND xPulsadorParo_NC
                        AND xReleTermico_NC
                        AND NOT xAlarmaRoturaCable;

(* ================================================================================
 * 2. NORMALIZACIÓN (NORM_X) Y ESCALADO LINEAL (SCALE_X) DE LAZO 4-20 mA
 *    Detección de rotura de hilo si iEntradaAnalogicaRaw < -1728 (< 3.0 mA)
 * ================================================================================ *)
IF (iEntradaAnalogicaRaw < -1728) OR (NOT xPulsadorParo_NC) OR (NOT xReleTermico_NC) THEN
    xAlarmaRoturaCable := TRUE;
    rValorNormalizado  := 0.0;
ELSE
    xAlarmaRoturaCable := FALSE;
    (* NORM_X: Convierte 0..27648 cuentas a rango flotante 0.0..1.0 *)
    rValorNormalizado  := INT_TO_REAL(iEntradaAnalogicaRaw) / 27648.0;
    IF rValorNormalizado < 0.0 THEN rValorNormalizado := 0.0; END_IF;
    IF rValorNormalizado > 1.0 THEN rValorNormalizado := 1.0; END_IF;
END_IF;

(* SCALE_X: Convierte 0.0..1.0 a Unidades de Ingeniería [rMinIngenieria .. rMaxIngenieria] *)
rVariableProcesoPV := rMinIngenieria + rValorNormalizado * (rMaxIngenieria - rMinIngenieria);

(* ================================================================================
 * 3. REGISTRO DE CORRIMIENTO DE BITS (SHRB / BSL) SINCRONIZADO CON ENCODER
 * ================================================================================ *)
xFlancoEncoder    := xPulsoEncoderBanda AND NOT xPulsoEncoderPrev;
xPulsoEncoderPrev := xPulsoEncoderBanda;

IF xFlancoEncoder AND xMotorBandaContactor THEN
    wRegistroCorrimiento := SHL(IN := wRegistroCorrimiento, N := 1);
    IF xCamaraPiezaDefecto THEN
        wRegistroCorrimiento := wRegistroCorrimiento OR 16#0001;
    END_IF;
END_IF;

(* Si el bit desplazado llega a la estación 5 (bit 4, máscara 16#0010), activa expulsor *)
xCilindroExpulsor := (wRegistroCorrimiento AND 16#0010) <> 16#0000;

(* ================================================================================
 * 4. SECUENCIADOR GRAFCET / SFC DE 4 ETAPAS CON TEMPORIZADOR TON Y CONTADOR CTU
 * ================================================================================ *)
IF NOT xMotorBandaContactor THEN
    iEstadoGrafcet := 0;
END_IF;

CASE iEstadoGrafcet OF
    0: (* Etapa 0: Reposo esperando pieza en estación *)
        xValvulaDosificadora := FALSE;
        IF xMotorBandaContactor AND xSensorPiezaEntrada THEN
            iEstadoGrafcet := 10;
        END_IF;
```

```
    10: (* Etapa 10: Dosificación temporizada con bloque TON *)
        xValvulaDosificadora := TRUE;
        IF tonDosificado.Q THEN
            xValvulaDosificadora := FALSE;
            iEstadoGrafcet       := 20;
        END_IF;

    20: (* Etapa 20: Incremento de contador de lote y espera de liberación *)
        xValvulaDosificadora := FALSE;
        IF NOT xSensorPiezaEntrada THEN
            iEstadoGrafcet := 0;
        END_IF;
END_CASE;

tonDosificado(IN := (iEstadoGrafcet = 10), PT := T#1500MS);
ctuContadorLotes(CU := (iEstadoGrafcet = 20), R := xResetContadores, PV := 50);
xLoteCompletado := ctuContadorLotes.Q;

(* ================================================================================
 * 5. BLOQUE TECNOLÓGICO PID DISCRETO CON ANTI-WINDUP Y DES-ESCALADO ANALÓGICO
 * ================================================================================ *)
IF xMotorBandaContactor AND NOT xAlarmaRoturaCable THEN
    VAR
        rErrorActual : REAL;
        rTerminoP    : REAL;
        rTerminoD    : REAL;
        rCandI       : REAL;
        rSumaUnsat   : REAL;
    END_VAR
    rErrorActual := rSetpointIngenieria - rVariableProcesoPV;
    rTerminoP    := rKp * rErrorActual;
    rTerminoD    := -rKp * rTd_s * (rVariableProcesoPV - rPvPrev) / rTs_s;
    rCandI       := rIntegralAcum + (rKp / rTi_s) * rTs_s * 0.5 * (rErrorActual + rErrorPrev);

    rSumaUnsat   := rTerminoP + rCandI + rTerminoD;
    IF rSumaUnsat > 100.0 THEN
        rSalidaControlPID := 100.0;
    ELSIF rSumaUnsat < 0.0 THEN
        rSalidaControlPID := 0.0;
    ELSE
        rSalidaControlPID := rSumaUnsat;
        rIntegralAcum     := rCandI;
    END_IF;

    rErrorPrev := rErrorActual;
    rPvPrev    := rVariableProcesoPV;
ELSE
    rSalidaControlPID := 0.0;
    rIntegralAcum     := 0.0;
END_IF;

(* Conversión de 0.0..100.0 % a 0..27648 cuentas crudas de salida analógica %QW80 *)
iSalidaAnalogicaRaw := REAL_TO_INT((rSalidaControlPID / 100.0) * 27648.0);

END_FUNCTION_BLOCK
```