CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA

Video 08 — Parte 4B: Interfases de Comunicación en Serie y Ejemplos de Acoplamiento Industrial mediante Interfase — UART, RS-232/RS-485 Diferencial, SPI, I2C, CAN Bus ISO 11898, Optoacopladores 24V, Drivers MOSFET/ULN2803 y SSR

Parte del Curso: Parte 4B — Interfases de Comunicación en Serie y Acoplamiento Industrial mediante Interfase
Arquitectura / Controlador Estudiado: Buses UART, RS-485 Diferencial (Modbus RTU), SPI, I2C, CAN Bus ISO 11898 y Etapas Optoacopladas 24V / MOSFET / SSR

```
R_opto = (24V - V_F) / I_F = (24 - 1.2) / 10mA = 2.28 kΩ | Lazo 4-20 mA sobre R_shunt = 165 Ω --> 0.66V a 3.30V
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Ventajas de la transmisión en serie frente a paralelo (eliminación de Clock Skew y Crosstalk); trama asíncrona UART 8N1 y par trenzado diferencial RS-485 (MAX485, terminadores de 120 Ω, 1200 m, protocolo Modbus RTU).
- Buses síncronos en placa y máquina: SPI Full-Duplex de 4 hilos (MOSI, MISO, SCLK, CS y los 4 modos CPOL/CPHA), I2C de 2 hilos en drenador abierto (SDA, SCL con Pull-Up, dirección de 7 bits y ACK) y CAN Bus ISO 11898 con arbitraje no destructivo por prioridad.
- Ejemplo 1 (Entrada 24V DC a MCU 3.3V): conexión de sensores industriales PNP (Sourcing) y NPN (Sinking) a través de optoacoplador (PC817/6N137) con resistencia limitadora, diodo antiparalelo y supresor TVS.
- Ejemplo 2 y 3 (Salidas DC, AC y Analógica): disparo de solenoides/motores de 24V DC con arreglo Darlington ULN2803 o MOSFET de nivel lógico + Diodo Flyback obligatorio; conmutación de potencia AC con Relé de Estado Sólido (SSR con Cruce por Cero y red Snubber RC) y receptor de 4-20 mA.

2. Parámetros Nominales de Configuración del Controlador (Video 08)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	4	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.8 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.15 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	10 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[4, 20]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[0, 250]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Comunicación en Serie frente a Paralelo en Sistemas Mecatrónicos

Eliminación del Desfase (Clock Skew) y Diafonía (Crosstalk), Modos Simplex/Half/Full-Duplex y Reloj Síncrono vs. Asíncrono

- **Turno 1 (Masculina):** En la Parte cuatro B estudiamos cómo se comunican los controladores con periféricos, sensores inteligentes y otros procesadores: ¿por qué las comunicaciones en serie reemplazaron casi por completo a los cables paralelos de ocho o dieciséis hilos?
- **Turno 2 (Femenina):** En un cable paralelo largo que intenta transmitir a alta frecuencia aparecen dos fenómenos físicos limitantes: la diafonía o Crosstalk por acoplamiento electromagnético entre hilos vecinos, y el desfase temporal o Clock Skew, porque los bits viajan a velocidades ligeramente distintas por cada alambre y llegan desalineados.
- **Turno 3 (Masculina):** La comunicación en serie envía un bit tras otro por un único canal a frecuencias mucho mayores, reduciendo drásticamente el costo del cableado, el tamaño de los conectores y el número de pines del microcontrolador.
- **Turno 4 (Femenina):** Clasificamos todo enlace serie según el sentido del flujo en Simplex de una sola vía, Half-Duplex bidireccional por turnos y Full-Duplex bidireccional simultáneo; y según su temporización en Síncrono, cuando incluye un hilo físico de reloj compartido, o Asíncrono, cuando cada trama lleva sus propios bits de sincronismo.

Diapositiva 02: Interfase Serie Universal Asíncrona (UART): Estructura de Trama

Línea Idle, Bit de Inicio (Start), 8 Bits de Datos (LSB primero), Bit de Paridad, Bits de Parada (Stop) y Baud Rate

- **Turno 1 (Masculina):** El protocolo serie más universal en sistemas embebidos es el Receptor y Transmisor Asíncrono Universal, conocido como UART, que comunica dos dispositivos en modo Full-Duplex cruzando únicamente la línea transmisora te equis con la receptora erre equis y compartiendo tierra.
- **Turno 2 (Femenina):** Al no existir hilo de reloj compartido, ambos extremos deben acordar previamente la misma velocidad en baudios o bits por segundo, como ciento quince mil doscientos baudios, con una diferencia de frecuencia entre sus osciladores menor al dos punto cinco por ciento.
- **Turno 3 (Masculina):** En la configuración estándar ocho ene uno, la línea permanece en reposo en nivel lógico alto; cuando el transmisor desea enviar un byte, baja la línea a cero durante un periodo de bit creando el Bit de Inicio o Start Bit, cuyo flanco de bajada despierta y sincroniza al reloj interno del receptor.
- **Turno 4 (Femenina):** A continuación el receptor muestrea en el centro exacto de cada pulso los ocho bits de datos comenzando por el bit menos significativo, evalúa el bit de paridad opcional para detectar errores de ruido, y verifica que la línea regrese a nivel alto en el Bit de Parada o Stop Bit.

Diapositiva 03: Estándares Físicos Serie: RS-232 frente a RS-422 y RS-485 Industrial

Tensión Bipolar (±12V con MAX232) frente a Señalización Diferencial en Par Trenzado (MAX485, 120 Ω y Modbus RTU)

- **Turno 1 (Masculina):** Un puerto UART directo del microcontrolador con niveles lógicos de cero a tres punto tres voltios solo es confiable dentro de la misma placa electrónica; si tendemos un cable de varios metros junto a un variador de frecuencia, el ruido inducido corromperá la trama por completo.
- **Turno 2 (Femenina):** El estándar clásico R S doscientos treinta y dos, implementado con el transceptor MAX doscientos treinta y dos, eleva la señal a voltajes bipolares de entre más menos cinco y más menos doce voltios respecto a tierra, alcanzando unos quince metros en enlaces punto a punto.
- **Turno 3 (Masculina):** Para cruzar una nave industrial completa de hasta mil doscientos metros de longitud y conectar hasta treinta y dos o doscientos cincuenta y seis equipos en un mismo bus, el estándar indiscutible es R S cuatrocientos ochenta y cinco usando transceptores como el MAX cuatrocientos ochenta y cinco.
- **Turno 4 (Femenina):** El secreto de R S cuatrocientos ochenta y cinco es la transmisión diferencial sobre un par de cables trenzados llamados A y B con resistencias terminadoras de ciento veinte ohmios en ambos extremos: el receptor solo mide la resta de voltaje entre el hilo A y el hilo B, cancelando de forma instantánea cualquier ruido electromagnético que afecte a ambos cables por igual.

Diapositiva 04: Interfase Periférica Serie Síncrona (SPI — Serial Peripheral Interface)

Topología Maestro-Esclavo de 4 Hilos (MOSI, MISO, SCLK, CS), Registro Circular y Modos de Reloj (CPOL y CPHA)

- **Turno 1 (Masculina):** Dentro de la placa de circuito impreso, cuando necesitamos comunicar al microcontrolador a velocidades de diez a ochenta megabits por segundo con conversores ADC rápidos, memorias Flash, tarjetas SD o pantallas gráficas, utilizamos la Interfase Periférica Serie o S P I.
- **Turno 2 (Femenina):** El bus S P I es síncrono, Full-Duplex y sigue una arquitectura Maestro-Esclavo de cuatro señales: la línea de reloj S C L K generada por el maestro, la línea M O S I del maestro al esclavo, la línea M I S O del esclavo al maestro, y una línea de Selección de Chip activa en bajo para cada esclavo.
- **Turno 3 (Masculina):** Su principio interno es sumamente eficiente: el registro del maestro y el del esclavo forman un anillo de desplazamiento circular de modo que, con cada ocho pulsos de reloj, el byte del maestro pasa al esclavo al mismo tiempo que el byte del esclavo entra al maestro.
- **Turno 4 (Femenina):** Al programar S P I es vital verificar en la hoja de datos del sensor cuál de los cuatro modos de reloj utiliza, definidos por la polaridad del reloj en reposo C P O L y por la fase de captura C P H A en el primer o segundo flanco.

Diapositiva 05: Bus de Circuito Inter-Integrado (I2C — Inter-Integrated Circuit)

Dos Hilos en Drenador Abierto (SDA y SCL), Resistencias Pull-Up, Direccionamiento de 7 Bits y Confirmación ACK/NACK

- **Turno 1 (Masculina):** ¿Qué hacemos cuando debemos conectar diez sensores distintos —un giroscopio inercial, un sensor de presión, un expansor de puertos y una pantalla OLED— pero no queremos gastar diez pines de selección de chip como en S P I? Utilizamos el bus I dos C creado por Philips.
- **Turno 2 (Femenina):** El bus I dos C conecta hasta ciento doce dispositivos compartiendo únicamente dos hilos: la línea de datos serie S D A y la línea de reloj serie S C L, operando a velocidades estándar de cien kilohercios, cuatrocientos kilohercios en modo rápido o hasta tres punto cuatro megahercios.
- **Turno 3 (Masculina):** Para que ningún dispositivo provoque un cortocircuito al hablar sobre los mismos dos cables, todos los transistores conectados a S D A y S C L son de Drenador Abierto: solo pueden tirar la línea hacia tierra, mientras que dos resistencias externas de Pull-Up, típicamente de dos punto dos o cuatro punto siete kiloohmios, devuelven la línea a nivel alto.
- **Turno 4 (Femenina):** En lugar de cables de selección físicos, cada transacción comienza con una condición de Inicio seguida de un byte que contiene la dirección única de siete bits del chip esclavo y el bit de lectura o escritura; y cada byte transmitido debe ser confirmado en el noveno pulso de reloj mediante un bit de reconocimiento ACK tirando la línea S D A a cero.

Diapositiva 06: Bus de Área de Controlador (CAN Bus ISO 11898) en Robótica y Automoción

Señalización Diferencial Dominante/Recesiva (CAN_H, CAN_L), Arbitraje No Destructivo por Prioridad y CRC en Hardware

- **Turno 1 (Masculina):** En automóviles modernos, brazos robóticos industriales, maquinaria agrícola y sistemas aeroespaciales, el bus en serie que interconecta todos los microcontroladores y servodrives es el Bus de Área de Controlador o CAN Bus, estandarizado bajo la norma ISO once mil ochocientos noventa y ocho.
- **Turno 2 (Femenina):** Al igual que R S cuatrocientos ochenta y cinco, la capa física de CAN utiliza un par trenzado diferencial con líneas CAN H y CAN L terminadas con resistencias de ciento veinte ohmios; pero define dos estados lógicos asimétricos: el estado Recesivo o uno lógico sin diferencia de tensión, y el estado Dominante o cero lógico con dos voltios de diferencia.
- **Turno 3 (Masculina):** En el bus CAN no existe un maestro central ni direcciones de nodos; las tramas se identifican por el contenido del mensaje mediante un Identificador de once o veintinueve bits, y si dos microcontroladores empiezan a transmitir al mismo instante se ejecuta el Arbitraje No Destructivo bit a bit.
- **Turno 4 (Femenina):** Como el cero es dominante y pisa al uno recesivo, el nodo cuyo identificador tenga ceros más significativos gana el acceso al bus sin que se pierda ni se corrompa ni un solo bit de su mensaje urgente; además, el hardware CAN verifica automáticamente quince bits de C R C, retransmite en caso de error y aísla nodos defectuosos.

Diapositiva 07: Ejemplo 1 de Acoplamiento de Entrada: Sensor Industrial de 24V a MCU de 3.3V

Sensores PNP (Sourcing) vs. NPN (Sinking), Aislamiento Galvánico con Optoacoplador (PC817/6N137) y Protección TVS

- **Turno 1 (Masculina):** Pasemos ahora a los tres ejemplos prácticos de diseño electrónico de interfases de acoplamiento: en el Ejemplo uno, ¿cómo conectamos un sensor inductivo o fotoeléctrico industrial de veinticuatro voltios al pin de tres punto tres voltios de un microcontrolador sin quemarlo ni meter ruido a su tierra?
- **Turno 2 (Femenina):** Primero identificamos el tipo de salida del sensor de tres hilos: un sensor P N P o Sourcing conmuta la carga hacia el positivo de veinticuatro voltios cuando detecta un objeto; mientras que un sensor N P N o Sinking conmuta el cable de señal hacia el negativo de cero voltios.
- **Turno 3 (Masculina):** Aunque un simple divisor resistivo bajaría el voltaje, dejaría unidas las tierras del campo industrial y del microcontrolador; el diseño profesional emplea un Optoacoplador como el P C ochocientos diecisiete o el rápido seis ene ciento treinta y siete, que transmite la señal mediante un haz de luz infrarroja a través de una barrera aislante de cinco mil voltios.
- **Turno 4 (Femenina):** Del lado de veinticuatro voltios colocamos una resistencia limitadora de dos punto dos kiloohmios para fijar unos diez miliamperios en el diodo emisor interno, un diodo en antiparalelo contra inversión de cables y un diodo supresor T V S; y del lado del fototransistor salimos limpiamente a tres punto tres voltios hacia el pin del microcontrolador.

Diapositiva 08: Ejemplo 2 de Acoplamiento de Salida DC: Cargas Inductivas con ULN2803 y MOSFET

Disparo de Electroválvulas, Relés y Solenoides de 24V, Transistores de Nivel Lógico y Diodo Flyback de Rueda Libre

- **Turno 1 (Masculina):** En el Ejemplo dos de acoplamiento debemos hacer el camino inverso: un pin del microcontrolador que apenas entrega tres punto tres voltios y diez miliamperios debe energizar una electroválvula neumática, un freno electromagnético o un contactor de veinticuatro voltios que consume varios amperios.
- **Turno 2 (Femenina):** Cuando necesitamos accionar hasta ocho relevadores o solenoides medianos de hasta quinientos miliamperios por canal, la interfase integrada clásica es el arreglo Darlington U L N veintiocho cero tres: recibe directamente señales de tres punto tres o cinco voltios y conmuta las cargas a tierra con diodos de protección ya integrados en el chip.
- **Turno 3 (Masculina):** Para corrientes mayores o modulación P W M de alta frecuencia sin caída de voltaje bipolar, diseñamos la etapa con un transistor MOSFET de Canal ene de Nivel Lógico —o precedido por un driver de compuerta—, colocando una resistencia de cien ohmios en serie con la compuerta y una de diez kiloohmios entre compuerta y fuente para mantenerlo apagado durante el arranque.
- **Turno 4 (Femenina):** Y jamás olvides el componente más crítico al conmutar cualquier bobina inductiva: el Diodo Flyback o de rueda libre conectado en antiparalelo directamente sobre los bornes de la válvula o motor; sin él, al apagarse el transistor la inductancia genera un pico inverso de cientos de voltios que perfora el silicio al instante.

Diapositiva 09: Ejemplo 3 de Acoplamiento AC y Analógico: Relé de Estado Sólido (SSR) y Lazo 4-20 mA

Opto-Triac con Detector de Cruce por Cero (MOC3041), Red Snubber RC y Conversión de Corriente 4-20 mA con Shunt

- **Turno 1 (Masculina):** En el Ejemplo tres abordamos dos interfases industriales esenciales: el control de cargas de potencia en corriente alterna de ciento veinte o doscientos veinte voltios, y la lectura de transmisores analógicos por lazo de corriente de cuatro a veinte miliamperios.
- **Turno 2 (Femenina):** Para conmutar resistencias calefactoras o motores de corriente alterna desde un controlador digital sin el desgaste ni el arco eléctrico de un contactor mecánico, utilizamos un Relé de Estado Sólido o S S R, construido internamente con un opto-triac aislador como el M O C treinta cuarenta y uno que dispara a un Triac de potencia.
- **Turno 3 (Masculina):** El circuito detector de Cruce por Cero integrado en el opto-triac asegura que la carga de corriente alterna se conecte únicamente en el instante exacto en que la onda senoidal pasa por cero voltios, eliminando los picos de corriente de irrupción y el ruido electromagnético, complementado con una red Snubber resistencia-capacitor en paralelo.
- **Turno 4 (Femenina):** Por último, para recibir señales analógicas desde sensores situados a decenas de metros usamos el estándar de cuatro a veinte miliamperios: como la corriente es idéntica en todo el lazo serie sin importar la resistencia del cable, en la entrada de nuestra interfase colocamos una resistencia Shunt de precisión de ciento sesenta y cinco ohmios que convierte cuatro a veinte miliamperios en un rango limpio de cero punto sesenta y seis a tres punto tres voltios, detectando además cualquier cable cortado si la corriente cae a cero.

Diapositiva 10: Síntesis de Comunicaciones en Serie y Acoplamiento por Interfase

Reglas de Compatibilidad Electromagnética (EMC), Separación de Tierras y Transición a los PLCs en la Parte 5A

- **Turno 1 (Masculina):** Con este módulo culminamos la Parte cuatro del curso, dominando tanto los protocolos de comunicación en serie como el diseño electrónico de interfases de entrada y salida de grado industrial.
- **Turno 2 (Femenina):** Sabemos cuándo utilizar UART, R S cuatrocientos ochenta y cinco diferencial con Modbus, S P I de alta velocidad, I dos C para múltiples sensores en placa o bus CAN con arbitraje por prioridad; y sabemos acoplar sensores de veinticuatro voltios, cargas inductivas DC, potencia AC y lazos de cuatro a veinte miliamperios.
- **Turno 3 (Masculina):** Ahora bien: ¿qué sucede cuando un fabricante empaqueta un microcontrolador robusto, todas estas interfases optoaisladas de veinticuatro voltios, puertos R S cuatrocientos ochenta y cinco y Ethernet, y una fuente industrial dentro de un chasis certificado para riel DIN?
- **Turno 4 (Femenina):** Nace el Controlador Lógico Programable; en la siguiente parte del curso estudiaremos a fondo la arquitectura, el ciclo de barrido y la programación industrial de los P L C bajo la norma internacional IEC sesenta y un mil ciento treinta y uno guion tres.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_08.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 08: Video 08 — Parte 4B: Interfases de Comunicación en Serie y Ejemplos de Acoplamiento Industrial
mediante Interfase
 * Archivo: controlador_tiempo_real_video_08.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:
 *   Buses UART, RS-485 Diferencial (Modbus RTU), SPI, I2C, CAN Bus ISO 11898 y Etapas Optoacopladas 24V / MOSFET
/ SSR
 * ============================================================================
 */

#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 08) ---
const float KP_GAIN      = 4.000f;
const float TI_SEC       = 0.800f;
const float TD_SEC       = 0.150f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0100f; // Periodo de muestreo Ts = 10 ms
const float U_MIN_LIM    = 4.0f;
const float U_MAX_LIM    = 20.0f;
const float PV_MIN_ENG   = 0.0f;
const float PV_MAX_ENG   = 250.0f;

// --- 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 08"));
  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_08.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
VIDEO 08: Video 08 — Parte 4B: Interfases de Comunicación en Serie y Ejemplos de Acoplamiento Industrial mediante
Interfase
Archivo: supervision_sbc_y_sintonizacion_video_08.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 08
KP_NOMINAL = 4
TI_NOMINAL = 0.8
TD_NOMINAL = 0.15
N_FILTRO   = 10
TS_SEG     = 0.0100
U_MIN      = 4
U_MAX      = 20


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: CONTROLADORES EN MECATRÓNICA — VIDEO 08")
    print(" Enfoque:", "Buses UART, RS-485 Diferencial (Modbus RTU), SPI, I2C, CAN Bus ISO 11898 y Etapas
Optoacopladas 24V / MOSFET / SSR")
    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=125)
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_08.st)

```
(* ================================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 08: Video 08 — Parte 4B: Interfases de Comunicación en Serie y Ejemplos de Acoplamiento Industrial
mediante Interfase
 * Archivo: automatizacion_plc_iec61131_video_08.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_V08) 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_V08
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 := 150.0;
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 := 250.0;

    (* 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 := 4.000;
    rTi_s                : REAL := 0.800;
    rTd_s                : REAL := 0.150;
```

```
    rTs_s               : REAL := 0.0100;
    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
```