CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA

Video 09 — Parte 5A: Controladores Lógicos Programables (PLC) — Estructura, Ciclo Scan, Escalera, IL, Enclavamiento y Secuenciación — Hardware Industrial Riel DIN, Entradas Sinking/Sourcing, Ciclo Scan (PII/PIQ), Norma IEC 61131-3, Ladder, IL y GRAFCET

Parte del Curso: Parte 5A — Controladores Lógicos Programables (PLC): Estructura, Scan, Escalera, IL y Secuenciación
Arquitectura / Controlador Estudiado: Arquitectura de Hardware PLC, Entradas Sinking/Sourcing, Ciclo Scan (PII/PIQ), Ladder, Lista de Instrucciones (IL) y GRAFCET

```
Enclavamiento: Q_Motor := (I_Marcha OR Q_Motor) AND I_Paro_NC AND I_Termico_NC | GRAFCET: E_k := (E_{k-1}·T_{k-1} + E_k)·NOT(E_{k+1})
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Estructura interna del PLC industrial en riel DIN (Fuente 24V DC, CPU con Watchdog, Memorias de Sistema/Programa/Datos Retentivos, bus Backplane y módulos DI/DO/AI/AO) y cableado de entradas Sinking (para sensores PNP) y Sourcing (para sensores NPN).
- Las 4 fases deterministas del Ciclo de Barrido (Scan Cycle): lectura de entradas físicas a la tabla de imagen PII (%I), ejecución secuencial del programa escribiendo en PIQ (%Q) y marcas (%M), autodiagnóstico/comunicaciones y volcado atómico a las salidas físicas.
- Norma internacional IEC 61131-3 y equivalencia exacta entre Diagrama de Escalera (contactos NO/NC, ramas serie AND, paralelo OR y bobinas directas/SET/RESET), Lista de Instrucciones (LD, LDN, AND, OR, ST, S, R) y Texto Estructurado (ST).
- Circuito de enclavamiento (Marcha-Paro con contacto de sello y pulsador de paro cableado físicamente como NC por seguridad ante rotura de hilo) y diseño sistemático de secuenciación industrial mediante GRAFCET / SFC (Etapas, Acciones y Transiciones).

2. Parámetros Nominales de Configuración del Controlador (Video 09)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	3	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	1.2 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.2 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	20 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[0, 100]	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: Origen y Rol del Controlador Lógico Programable (PLC) en Mecatrónica

Sustitución de Tableros de Relevadores Electromecánicos, Inmunidad al Ruido, Rango Térmico y Montaje en Riel DIN

- **Turno 1 (Masculina):** Bienvenidos a la Parte cinco del curso, dedicada al pilar indiscutible de la automatización industrial y las celdas de manufactura mecatrónica en todo el planeta: el Controlador Lógico Programable o P L C.
- **Turno 2 (Femenina):** Antes de finales de la década de mil novecientos sesenta, el control secuencial de una línea automotriz requería armarios gigantescos con miles de relevadores electromecánicos y temporizadores cableados a mano; cualquier cambio en el proceso obligaba a detener la fábrica durante semanas para recablear todo el tablero.
- **Turno 3 (Masculina):** El P L C nació como una computadora industrial diseñada específicamente para reemplazar esa lógica cableada por un programa en memoria, soportando temperaturas extremas, vibración mecánica, polvo y severas interferencias electromagnéticas sin fallar durante décadas de operación continua.
- **Turno 4 (Femenina):** A diferencia de una tarjeta electrónica desnuda, un P L C ofrece bornes robustos a veinticuatro voltios, aislamiento óptico integrado en cada canal, indicadores LED de diagnóstico en cada entrada y salida, expansión modular en riel DIN y programación estandarizada comprensible por todo el personal de planta.

Diapositiva 02: Estructura Física e Interna del PLC (Compacto frente a Modular)

Fuente de Alimentación, CPU, Memorias (Sistema, Programa y Datos Retentivos), Bus Backplane y Módulos de E/S

- **Turno 1 (Masculina):** Todo Controlador Lógico Programable, ya sea un modelo compacto todo-en-uno para máquinas pequeñas o un sistema modular en bastidor para plantas completas, se organiza en cinco bloques funcionales internos.
- **Turno 2 (Femenina):** El primer bloque es la Fuente de Alimentación, que convierte la red eléctrica o los veinticuatro voltios externos en las tensiones reguladas y protegidas que alimentan al bus interno; el segundo bloque es la Unidad Central de Procesamiento o CPU, que incluye un procesador de alta confiabilidad y un circuito supervisor Watchdog.
- **Turno 3 (Masculina):** El tercer bloque es el Sistema de Memorias, dividido en la memoria ROM de firmware del fabricante, la memoria Flash de programa del usuario y la memoria RAM de datos con respaldo no volátil para conservar contadores, recetas y tablas de imagen ante cortes de energía.
- **Turno 4 (Femenina):** Completan la arquitectura el bus de fondo o Backplane de alta velocidad que comunica la CPU con las tarjetas laterales, y los Módulos de Entrada y Salida digitales, analógicos, de conteo rápido y de redes industriales como PROFINET, EtherNet IP, Modbus TCP y EtherCAT.

Diapositiva 03: Arquitecturas de Entradas y Salidas Digitales de un PLC

Entradas Sumidero (Sinking) frente a Fuente (Sourcing) y Salidas a Relé (Secas), Transistor (24V DC Rápidas) y Triac

- **Turno 1 (Masculina):** Para cablear un tablero de control sin dañar módulos ni cometer errores de polaridad, debemos dominar las configuraciones eléctricas de las tarjetas de entradas y salidas digitales del P L C.
- **Turno 2 (Femenina):** En los módulos de entrada de veinticuatro voltios existen dos topologías según cómo se conecte el borne común de los optoacopladores internos: en una Entrada Tipo Sumidero o Sinking el borne común se conecta a cero voltios, por lo que la corriente entra al P L C desde sensores tipo P N P.
- **Turno 3 (Masculina):** Inversamente, en una Entrada Tipo Fuente o Sourcing el borne común se conecta al positivo de veinticuatro voltios, de modo que la corriente sale del borne del P L C hacia sensores tipo N P N que cierran el circuito a tierra.
- **Turno 4 (Femenina):** En cuanto a los módulos de salida digital, elegimos entre tres tecnologías: salidas a Relé con contactos secos libres de potencial que conmutan tanto corriente continua como alterna pero son lentas y tienen desgaste mecánico; salidas a Transistor de veinticuatro voltios con vida ilimitada y velocidad de kilohercios para servomotores; y salidas a Triac exclusivas para cargas de corriente alterna.

Diapositiva 04: Procesamiento Determinista: El Ciclo de Barrido (Scan Cycle)

Lectura a Tabla de Imagen de Entradas (PII), Ejecución de Programa, Diagnóstico y Escritura a Imagen de Salidas (PIQ)

- **Turno 1 (Masculina):** Una de las diferencias conceptuales más importantes entre un programa convencional de computadora y un P L C es la forma en que se ejecuta la lógica y se actualizan los pines: el P L C opera mediante un bucle determinista de cuatro fases llamado Ciclo de Barrido o Scan Cycle.
- **Turno 2 (Femenina):** En la Fase uno, antes de ejecutar una sola línea de lógica, la CPU lee simultáneamente todos los bornes físicos de entrada y guarda una fotografía congelada de su estado en una zona de memoria RAM llamada Tabla de Imagen de Proceso de Entradas, abreviada P I I.
- **Turno 3 (Masculina):** En la Fase dos, la CPU ejecuta el programa del usuario desde el primer peldaño hasta el último: durante toda esa evaluación jamás vuelve a mirar los bornes físicos, sino que lee de la tabla P I I y escribe los nuevos resultados en la Tabla de Imagen de Proceso de Salidas, abreviada P I Q.
- **Turno 4 (Femenina):** En la Fase tres el P L C atiende las comunicaciones con pantallas H M I y realiza su autodiagnóstico de seguridad; y en la Fase cuatro vuelca de golpe toda la tabla P I Q hacia los bornes físicos de salida, garantizando que ningún sensor pueda cambiar a mitad de cálculo y provocar condiciones de carrera.

Diapositiva 05: Estándar Internacional IEC 61131-3 y los 5 Lenguajes de PLC

Unidades de Organización (PRG, FB, FC), Diagrama de Escalera (LD), Bloques (FBD), Lista de Instrucciones (IL), ST y SFC

- **Turno 1 (Masculina):** Durante décadas cada fabricante de P L C tenía su propia sintaxis incompatible, hasta que la Comisión Electrotécnica Internacional promulgó la norma IEC sesenta y un mil ciento treinta y uno guion tres, que unificó la programación industrial en todo el mundo.
- **Turno 2 (Femenina):** Esta norma estructura el software en tres tipos de Unidades de Organización de Programa: los Programas principales, las Funciones sin memoria interna, y los Bloques de Función que poseen su propia base de datos de instancia para recordar su estado de un ciclo Scan al siguiente.
- **Turno 3 (Masculina):** Asimismo, la norma define cinco lenguajes oficiales que pueden combinarse dentro de un mismo proyecto: tres lenguajes gráficos —que son el Diagrama de Escalera o Ladder, el Diagrama de Bloques de Función y el Gráfico Funcional Secuencial o GRAFCET—.
- **Turno 4 (Femenina):** Y dos lenguajes textuales: la Lista de Instrucciones, similar a un ensamblador lógico basado en acumulador, y el Texto Estructurado, un lenguaje de alto nivel similar a Pascal y C ideal para fórmulas matemáticas, bucles, arreglos y máquinas de estados complejas.

Diapositiva 06: Programación en Escalera (Ladder Diagram): Contactos, Bobinas y Lógica

Rieles de Alimentación, Peldaños (Rungs), Contactos NO y NC, Compuertas AND/OR y Bobinas Directas, SET y RESET

- **Turno 1 (Masculina):** El lenguaje de programación en Escalera o Ladder Diagram sigue siendo el más utilizado en mantenimiento y automatización porque representa el programa como un plano eléctrico vertical con dos rieles laterales unidos por peldaños horizontales.
- **Turno 2 (Femenina):** A la izquierda de cada peldaño colocamos las condiciones de entrada representadas como contactos: un contacto Normalmente Abierto deja pasar la continuidad lógica cuando su bit en memoria vale uno; mientras que un contacto Normalmente Cerrado conduce cuando su bit en memoria vale cero.
- **Turno 3 (Masculina):** Al conectar dos contactos en serie sobre el mismo peldaño construimos una operación lógica AND, pues ambos deben conducir para energizar el extremo derecho; y al colocar dos ramas de contactos en paralelo construimos una operación lógica OR.
- **Turno 4 (Femenina):** En el extremo derecho del peldaño se sitúan las salidas o Bobinas: la bobina normal escribe un uno mientras el peldaño sea verdadero y vuelve a cero en cuanto cae la condición; mientras que la bobina SET engancha el bit en uno de forma permanente hasta que otro peldaño active su bobina RESET correspondiente.

Diapositiva 07: Programación en Lista de Instrucciones (IL) y Texto Estructurado (ST)

Operadores de Acumulador (LD, LDN, AND, OR, ST, S, R) y Equivalencia Exacta Peldaño a Peldaño con Escalera

- **Turno 1 (Masculina):** Comprender cómo se traduce un peldaño de Escalera a Lista de Instrucciones y a Texto Estructurado nos permite dominar la lógica interna con la que el compilador del P L C evalúa cualquier red de contactos.
- **Turno 2 (Femenina):** En Lista de Instrucciones todo peldaño comienza cargando el primer contacto en el registro acumulador de resultado mediante el comando L D si el contacto es normalmente abierto, o L D N si es normalmente cerrado.
- **Turno 3 (Masculina):** A continuación, cada contacto en serie se combina con la instrucción AND o AND N, y cada rama en paralelo se une con la instrucción OR u OR N, utilizando paréntesis cuando una rama paralela contiene varios contactos en serie.
- **Turno 4 (Femenina):** Finalmente, la instrucción S T almacena el valor del acumulador en la bobina de salida, o bien los comandos S y R ejecutan el enganche Set o Reset; y en Texto Estructurado toda esa secuencia se escribe en una sola línea clara de asignación booleana con operadores AND, OR y NOT.

Diapositiva 08: Circuito de Enclavamiento (Marcha-Paro) y Relevadores Internos (%M)

Contacto de Sello en Paralelo (Self-Holding), Seguridad de Cableado NC en el Paro y Uso de Marcas Auxiliares de Memoria

- **Turno 1 (Masculina):** El bloque constructivo más célebre de la ingeniería de control con P L C es el circuito de Enclavamiento o auto-retención de Marcha y Paro, empleado para arrancar motores y secuencias con pulsadores momentáneos que regresan por resorte.
- **Turno 2 (Femenina):** Cuando el operador presiona por un instante el botón de Marcha, se energiza la bobina del contactor del motor; pero para que el motor no se apague al soltar el dedo, colocamos en paralelo con el botón de Marcha un contacto normalmente abierto asociado a la propia bobina del motor, llamado contacto de sello o retención.
- **Turno 3 (Masculina):** En serie con esa rama paralela colocamos las condiciones de detención: el pulsador de Paro y el contacto auxiliar del relé térmico de sobrecarga; aquí aplica una regla de seguridad industrial obligatoria: en el tablero físico, todo botón de Paro debe cablearse usando un contacto físico Normalmente Cerrado.
- **Turno 4 (Femenina):** Al cablear el Paro como normalmente cerrado en campo, el borne de entrada del P L C recibe veinticuatro voltios permanentes mientras todo está bien, y si se presiona el botón o se rompe accidentalmente el cable en la máquina, la señal cae a cero deteniendo el motor de manera segura; además, usamos Marcas internas de memoria M para crear enclavamientos auxiliares sin gastar salidas físicas.

Diapositiva 09: Diseño Sistemático de Secuenciación Industrial y GRAFCET (SFC)

Método Paso a Paso con Enclavamiento y Gráfico Funcional de Etapas, Acciones y Transiciones (Norma IEC 60848)

- **Turno 1 (Masculina):** Cuando automatizamos una estación mecatrónica con múltiples cilindros neumáticos y sensores magnéticos —por ejemplo, sujetar pieza, bajar taladro, subir taladro y soltar pieza—, programar peldaños por intuición conduce a bloqueos cuando dos sensores coinciden activados.
- **Turno 2 (Femenina):** La metodología formal internacional para diseñar cualquier automatismo secuencial sin errores es el diagrama GRAFCET, incorporado en los P L C como lenguaje de Gráfico Funcional Secuencial o S F C.
- **Turno 3 (Masculina):** Un GRAFCET modela la máquina como una sucesión ordenada de Etapas representadas por cuadrados numerados, comenzando por la Etapa Inicial cero de reposo, separadas por líneas horizontales llamadas Transiciones que tienen asociada una condición lógica de sensores llamada Receptividad.
- **Turno 4 (Femenina):** Su traducción al lenguaje de Escalera mediante el método paso a paso es exacta y universal: cada etapa se representa con una marca interna de memoria que se activa cuando la etapa anterior está activa y se cumple la condición de transición, y esa misma etapa recién activada apaga de inmediato a la etapa anterior.

Diapositiva 10: Síntesis de Arquitectura PLC, Escalera, Lista de Instrucciones y Secuenciación

Buenas Prácticas de Programación IEC 61131-3 y Transición hacia la Programación Avanzada de PLC en la Parte 5B

- **Turno 1 (Masculina):** En esta Parte cinco A hemos construido los cimientos sólidos de la ingeniería con Controladores Lógicos Programables: desde su estructura de hardware industrial y el cableado de entradas Sinking y Sourcing, hasta las cuatro fases del ciclo de barrido Scan.
- **Turno 2 (Femenina):** Dominamos el estándar IEC sesenta y un mil ciento treinta y uno guion tres, la programación en Escalera, su equivalencia directa con Lista de Instrucciones y Texto Estructurado, el circuito seguro de enclavamiento y el diseño sistemático de secuencias con GRAFCET.
- **Turno 3 (Masculina):** Recuerda siempre la regla de higiene en Escalera: nunca coloques dos bobinas ordinarias dirigidas a la misma salida física en dos peldaños distintos, porque debido al ciclo Scan solo prevalecerá el estado del último peldaño evaluado.
- **Turno 4 (Femenina):** En el siguiente módulo, correspondiente a la Parte cinco B, completaremos el dominio del P L C estudiando los temporizadores, contadores, registros de corrimiento, relés maestros de control, saltos, manejo numérico de datos y procesamiento de entradas y salidas analógicas.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_09.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 09: Video 09 — Parte 5A: Controladores Lógicos Programables (PLC) — Estructura, Ciclo Scan, Escalera, IL, Enclavamiento y Secuenciación
 * Archivo: controlador_tiempo_real_video_09.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:
 *   Arquitectura de Hardware PLC, Entradas Sinking/Sourcing, Ciclo Scan (PII/PIQ), Ladder, Lista de Instrucciones (IL) y GRAFCET
 * ============================================================================
 */

#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 09) ---
const float KP_GAIN      = 3.000f;
const float TI_SEC       = 1.200f;
const float TD_SEC       = 0.200f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0200f; // Periodo de muestreo Ts = 20 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 100.0f;
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 09"));
  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_09.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
VIDEO 09: Video 09 — Parte 5A: Controladores Lógicos Programables (PLC) — Estructura, Ciclo Scan, Escalera, IL,
Enclavamiento y Secuenciación
Archivo: supervision_sbc_y_sintonizacion_video_09.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 09
KP_NOMINAL = 3
TI_NOMINAL = 1.2
TD_NOMINAL = 0.2
N_FILTRO   = 10
TS_SEG     = 0.0200
U_MIN      = 0
U_MAX      = 100


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 09")
    print(" Enfoque:", "Arquitectura de Hardware PLC, Entradas Sinking/Sourcing, Ciclo Scan (PII/PIQ), Ladder,
Lista de Instrucciones (IL) y GRAFCET")
    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_09.st)

```
(* ==========================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 09: Video 09 — Parte 5A: Controladores Lógicos Programables (PLC) — Estructura, Ciclo Scan, Escalera, IL,
Enclavamiento y Secuenciación
 * Archivo: automatizacion_plc_iec61131_video_09.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_V09) 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_V09
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 := 3.000;
    rTi_s                : REAL := 1.200;
    rTd_s                : REAL := 0.200;
```

```
    rTs_s                : REAL := 0.0200;
    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
```