CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA

Video 10 — Parte 5B: Programación Avanzada de PLC — Temporizadores, Contadores, Registros, MCR, Salto, Datos y E/S Analógica — Temporizadores TON/TOF/TP/TONR, Contadores CTU/CTD/HSC, Shift Registers, MCR, JMP/LBL, Comparadores y Escalado 4-20 mA

Parte del Curso: Parte 5B — Programación Avanzada de PLC: Temporizadores, Contadores, Registros, MCR y Analógicas
Arquitectura / Controlador Estudiado: Bloques IEC 61131-3 (TON/TOF/TP/TONR, CTU/CTD/HSC), Shift Registers (SHRB), Zona MCR, Saltos, Aritmética y Escalado 4-20 mA

```
u_norm = NORM_X(0, 27648, AI_Raw) = AI_Raw / 27648.0 | PV_Eng = SCALE_X(Y_min, Y_max, u_norm) = Y_min + u_norm·(Y_max - Y_min)
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Temporizadores estándar (TON retardo a la conexión, TOF retardo a la desconexión, TP pulso y TONR acumulador retentivo) y Contadores (CTU, CTD, CTUD por flanco y Contadores de Alta Velocidad HSC por hardware para encoders).
- Registros de corrimiento de bits (Shift Registers SHRB/BSL) sincronizados con el encoder de banda transportadora para el seguimiento y expulsión exacta de productos defectuosos aguas abajo.
- Control de Relé Maestro (zonas MCR_ON / MCR_OFF que fuerzan a cero bobinas no retentivas) frente al peligro de bobinas congeladas en saltos condicionales (JMP / LBL), y modularización con Bloques de Función (FB con DB de instancia).
- Tipos de datos (BOOL, BYTE, WORD/INT 16-bit, DWORD/DINT/REAL 32-bit), transferencia MOV, comparadores relacionales, bloques aritméticos y escalado lineal de entradas analógicas de 0-10V y 4-20 mA (0..27648 cuentas) con NORM_X y SCALE_X.

2. Parámetros Nominales de Configuración del Controlador (Video 10)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	4.5	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	1.5 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.25 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	50 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[0, 27648]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[-50, 150]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Temporizadores en PLC bajo Norma IEC 61131-3 (TON, TOF, TP y TONR)

Entradas de Habilitación (IN) y Preselección (PT), Salidas de Estado (Q) y Tiempo Transcurrido (ET) en Automatización

- **Turno 1 (Masculina):** En la Parte cinco B abordamos los bloques funcionales avanzados que dotan a un P L C de control temporal, conteo, seguimiento de piezas, aritmética y control continuo, comenzando por los cuatro tipos de Temporizadores estandarizados.
- **Turno 2 (Femenina):** Todo bloque temporizador recibe una entrada booleana de habilitación IN y un tiempo preseleccionado P T, y entrega una salida booleana cu junto con el tiempo transcurrido actual E T: el más utilizado es el temporizador con retardo a la conexión, abreviado T O N.
- **Turno 3 (Masculina):** En un temporizador T O N, cuando la entrada IN pasa a verdadero el reloj empieza a contar, pero la salida cu solo se enciende cuando la entrada se ha mantenido verdadera de forma ininterrumpida durante todo el tiempo preseleccionado; si la entrada cae antes, el conteo vuelve a cero al instante.
- **Turno 4 (Femenina):** Lo complementan el temporizador con retardo a la desconexión T O F, que mantiene encendido un ventilador o bomba de lubricación durante un tiempo fijo después de haberse apagado la máquina; el generador de pulso exacto T P; y el temporizador retentivo T O N R, que acumula horas totales de operación para avisos de mantenimiento.

Diapositiva 02: Contadores en PLC (CTU, CTD, CTUD) y Contadores de Alta Velocidad (HSC)

Detección de Flancos, Valor Preseleccionado (PV), Acumulado (CV), Límite del Ciclo Scan y Hardware HSC para Encoders

- **Turno 1 (Masculina):** Para contabilizar botellas envasadas, ciclos de prensado o lotes de producción utilizamos los bloques Contadores del P L C, los cuales incrementan o decrementan su variable interna únicamente al detectar un flanco de subida en su entrada de pulsos.
- **Turno 2 (Femenina):** El Contador Ascendente C T U suma uno en su registro acumulado C V con cada pieza detectada, activa su salida cu cuando el conteo iguala o supera al valor preseleccionado P V, y vuelve a cero al activar su entrada de reinicio Reset.
- **Turno 3 (Masculina):** Por su parte, el Contador Descendente C T D parte cargando el valor inicial mediante la entrada Load y descuenta hacia cero; mientras que el Contador Bidireccional C T U D combina una entrada de suma y otra de resta para llevar en tiempo real el inventario exacto de piezas dentro de un almacén intermedio.
- **Turno 4 (Femenina):** Un detalle crucial de ingeniería: un contador normal por software no puede contar pulsos más rápidos que la mitad de la frecuencia del ciclo Scan; cuando conectamos un encoder rotatorio que entrega decenas de miles de pulsos por segundo, debemos habilitar en la configuración de hardware del P L C un Contador de Alta Velocidad o H S C.

Diapositiva 03: Registros de Corrimiento (Shift Registers — SHRB, BSL, BSR)

Desplazamiento Sincronizado de Bits para Seguimiento y Rechazo de Piezas Defectuosas en Bandas Transportadoras

- **Turno 1 (Masculina):** Imagina una línea embotelladora de alta velocidad donde una cámara de visión artificial detecta una tapa mal colocada en la estación uno, pero el cilindro neumático expulsor se encuentra cinco botellas más adelante sobre la banda transportadora: ¿cómo sabe el P L C cuándo activar el expulsor?
- **Turno 2 (Femenina):** Este problema clásico de rastreo de productos en movimiento se resuelve de manera impecable utilizando una instrucción de Registro de Corrimiento de Bits, conocida como Shift Register o desplazamiento de bits a la izquierda o derecha.
- **Turno 3 (Masculina):** El bloque de registro de corrimiento mantiene una fila de bits en memoria que representa las posiciones físicas sobre la banda: tiene una entrada de datos donde la cámara inyecta un uno si la pieza es defectuosa o un cero si es buena, y una entrada de reloj conectada al sensor de paso de la banda.
- **Turno 4 (Femenina):** Cada vez que la banda avanza una posición, todos los bits de la fila se desplazan un casillero hacia adelante; así, el bit uno que nació en la cámara viaja sincronizado con la botella física hasta llegar exactamente al bit número cinco, el cual dispara la electroválvula de rechazo sin confundir jamás una pieza con otra.

Diapositiva 04: Controles de Relé Maestro (MCR — Master Control Relay)

Zonas de Habilitación Condicional de Peldaños por Software y Diferencia Crítica con el Paro de Emergencia Cableado

- **Turno 1 (Masculina):** En programas de Escalera extensos a menudo necesitamos habilitar o deshabilitar simultáneamente toda una sección de veinte o treinta peldaños correspondientes a una estación de trabajo según el modo de operación seleccionado.
- **Turno 2 (Femenina):** Para no repetir el mismo contacto de condición al inicio de cada uno de los treinta peldaños, utilizamos las instrucciones de Control de Relé Maestro o M C R, que delimitan una zona de programa entre un peldaño de apertura M C R ON y un peldaño de cierre M C R OFF.
- **Turno 3 (Masculina):** Mientras la condición del peldaño maestro M C R sea verdadera, todos los peldaños dentro de la zona se evalúan con normalidad; pero si la condición maestra pasa a falso, el P L C fuerza automáticamente a cero todas las bobinas no retentivas y reinicia los temporizadores T O N situados dentro de esa zona.
- **Turno 4 (Femenina):** Debemos subrayar una norma vital de seguridad funcional: una zona M C R de software jamás sustituye al Relevador Maestro de Seguridad físico cableado en el tablero; ante un Paro de Emergencia, la ley exige cortar físicamente la energía eléctrica de los actuadores mediante un relé de seguridad certificado.

Diapositiva 05: Instrucciones de Control de Salto (JMP / LBL) y Subrutinas (FC / FB)

Alteración del Flujo Secuencial del Scan, Peligro de Bobinas Congeladas y Modularización con Bloques de Instancia

- **Turno 1 (Masculina):** A primera vista, la instrucción de Salto condicional J M P hacia una etiqueta L B L parece similar a una zona de Relé Maestro M C R, pero su efecto sobre las salidas del P L C es radicalmente distinto y requiere sumo cuidado.
- **Turno 2 (Femenina):** En una zona M C R desactivada, el P L C sí recorre los peldaños escribiendo ceros en las bobinas; en cambio, cuando se activa una instrucción de salto J M P, la CPU salta por encima de esos peldaños sin evaluarlos en absoluto durante ese ciclo Scan.
- **Turno 3 (Masculina):** Eso significa que todas las salidas digitales, marcas y temporizadores que quedaron dentro de la región saltada quedan congelados en el último estado que tenían antes del salto: si un motor estaba encendido en uno dentro de la zona saltada, ¡permanecerá encendido indefinidamente aunque cambien sus sensores!
- **Turno 4 (Femenina):** Por ello en la programación moderna bajo norma IEC preferimos estructurar el código mediante llamadas a Funciones F C y Bloques de Función F B parametrizados: escribimos y probamos una sola vez el bloque de control de un motor o válvula, y lo instanciamos veinte veces con su propio bloque de datos.

Diapositiva 06: Tipos y Manejo de Datos en PLC e Instrucciones de Transferencia (MOV)

Jerarquía de Bits (BOOL), Bytes, Palabras de 16 Bits (WORD/INT) y Dobles Palabras de 32 Bits (DWORD/DINT/REAL)

- **Turno 1 (Masculina):** Un P L C moderno es mucho más que un procesador de bits booleanos; para manejar recetas, conteos, velocidades y temperaturas debemos dominar la organización de su memoria en registros numéricos y las instrucciones de manejo de datos.
- **Turno 2 (Femenina):** La jerarquía de datos parte de la unidad mínima booleana BOOL de un bit; ocho bits consecutivos forman un BYTE; dos bytes forman una Palabra de dieciséis bits que puede interpretarse como patrón binario WORD o como entero con signo INT desde menos treinta y dos mil setecientos sesenta y ocho hasta más treinta y dos mil setecientos sesenta y siete.
- **Turno 3 (Masculina):** Cuatro bytes consecutivos conforman una Doble Palabra de treinta y dos bits, empleada para enteros largos D I N T o para números en punto flotante de precisión simple tipo REAL, indispensables para cálculos físicos con decimales.
- **Turno 4 (Femenina):** La instrucción fundamental para manipular estos registros es la transferencia MOVE o M O V, que al activarse su peldaño copia el valor numérico de una constante o registro origen hacia un registro destino —permitiendo, por ejemplo, cambiar instantáneamente el tiempo preseleccionado de un temporizador o la velocidad de una receta—.

Diapositiva 07: Instrucciones de Comparación y Bloques Aritméticos en Línea

Comparadores Relacionales (==, <>, >, <, >=, <=, IN_RANGE), Operaciones (ADD, SUB, MUL, DIV) y Conversión de Tipos

- **Turno 1 (Masculina):** Una vez que tenemos variables numéricas enteras o en punto flotante dentro de la memoria del P L C, ¿cómo hacemos que un peldaño de Escalera tome decisiones lógicas basadas en esos números? Mediante las instrucciones de Comparación.
- **Turno 2 (Femenina):** Los comparadores se insertan en el peldaño exactamente igual que un contacto: evalúan si el operando A es igual, diferente, mayor, menor, mayor o igual, o menor o igual que el operando B; si la relación matemática se cumple, el bloque conduce la continuidad lógica hacia la bobina de salida.
- **Turno 3 (Masculina):** Los entornos actuales incluyen además los bloques comparadores de ventana IN RANGE y OUT RANGE, ideales para verificar en un solo paso si la presión o el peso de un producto se encuentra dentro de las tolerancias de calidad mínima y máxima.
- **Turno 4 (Femenina):** Junto a los comparadores disponemos de los bloques aritméticos de suma ADD, resta SUB, multiplicación MUL, división DIV y bloque de fórmula general CALCULATE, recordando siempre convertir previamente los enteros a tipo REAL para no perder los decimales en las divisiones.

Diapositiva 08: Módulos de Entrada y Salida Analógica en PLC

Estándares de Tensión (0-10V, ±10V) y Corriente (4-20 mA), Transmisores de 2, 3 y 4 Hilos y Cuentas del ADC (0 a 27648)

- **Turno 1 (Masculina):** Para que el P L C pueda medir variables físicas continuas como temperatura, presión, nivel, caudal o posición lineal y comandar variadores de frecuencia o válvulas proporcionales, incorporamos módulos de Entrada y Salida Analógica.
- **Turno 2 (Femenina):** En la industria existen dos estándares eléctricos dominantes para señales analógicas: las señales de tensión de cero a diez voltios o más menos diez voltios, empleadas dentro del mismo gabinete eléctrico; y los lazos de corriente de cero a veinte o de cuatro a veinte miliamperios para sensores distribuidos en planta.
- **Turno 3 (Masculina):** Al cablear un transmisor de presión de cuatro a veinte miliamperios de dos hilos, el positivo de veinticuatro voltios va al borne positivo del sensor, y la salida negativa del sensor entra al canal analógico positivo del P L C para cerrar el lazo a tierra a través de la resistencia interna del módulo.
- **Turno 4 (Femenina):** El convertidor analógico-digital del módulo del P L C transforma esa corriente o voltaje en un número entero crudo de dieciséis bits: por ejemplo, en la plataforma industrial Siemens ese rango nominal de cuatro a veinte miliamperios o de cero a diez voltios se representa exactamente entre cero y veintisiete mil seiscientas cuarenta y ocho cuentas.

Diapositiva 09: Normalización y Escalado Matemático de Señales Analógicas (NORM_X y SCALE_X)

Conversión Lineal de Cuentas Crudas del ADC a Unidades de Ingeniería (°C, bar, RPM) y Alarma de Rotura de Hilo

- **Turno 1 (Masculina):** Cuando el módulo de entrada analógica nos entrega un número entero como trece mil ochocientos veinticuatro cuentas en el registro de entrada, el operador y nuestro algoritmo de control necesitan saber a cuántos bares de presión o grados Celsius equivale ese número: este proceso se llama Escalado Lineal.
- **Turno 2 (Femenina):** En la programación moderna en Escalera o Texto Estructurado realizamos esta conversión encadenando dos funciones estándar: primero la instrucción de Normalización NORM equis, y segundo la instrucción de Escalado SCALE equis.
- **Turno 3 (Masculina):** La función NORM equis toma el entero crudo de entrada —entre cero y veintisiete mil seiscientas cuarenta y ocho cuentas— y lo divide linealmente para entregar un número en punto flotante normalizado entre cero punto cero y uno punto cero, que representa el porcentaje de la señal.
- **Turno 4 (Femenina):** Acto seguido, la función SCALE equis multiplica ese valor normalizado por el intervalo de unidades de ingeniería del sensor —por ejemplo, de menos cincuenta a más ciento cincuenta grados Celsius— entregando la variable física real lista para compararse o entrar a un bloque pe i de; y si las cuentas caen a un valor negativo fuerte, disparamos de inmediato la alarma de cable cortado.

Diapositiva 10: Integración Total en un PLC Industrial (Rumbo al Cierre del Curso)

Combinación de Secuencia GRAFCET, Temporizadores/Contadores, Escalado Analógico y Bloque Tecnológico PID_Compact

- **Turno 1 (Masculina):** Culminamos la Parte cinco y los diez módulos técnicos del curso integrando todas las herramientas en una arquitectura real de automatización industrial dentro del P L C.
- **Turno 2 (Femenina):** En una planta mecatrónica completa, la secuencia maestra diseñada en GRAFCET gobierna las etapas de llenado, dosificación y vaciado utilizando enclavamientos, temporizadores T O N, contadores de lotes y registros de corrimiento.
- **Turno 3 (Masculina):** De manera simultánea, un bloque de interrupción cíclica de tiempo ejecuta cada cincuenta milisegundos la lectura de los transmisores de cuatro a veinte miliamperios escalados con NORM equis y SCALE equis, alimentando al bloque tecnológico pe i de integrado en el P L C para regular la temperatura y la velocidad de los variadores.
- **Turno 4 (Femenina):** Con este dominio integral desde el lazo cerrado continuo y los microcontroladores hasta las interfases y los P L C, estamos listos para pasar al cierre del curso donde sintetizaremos todo nuestro recorrido de ingeniería.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_10.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 10: Video 10 — Parte 5B: Programación Avanzada de PLC — Temporizadores, Contadores, Registros, MCR,
Salto, Datos y E/S Analógica
 * Archivo: controlador_tiempo_real_video_10.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:
 *   Bloques IEC 61131-3 (TON/TOF/TP/TONR, CTU/CTD/HSC), Shift Registers (SHRB), Zona MCR, Saltos, Aritmética y
Escalado 4-20 mA
 * ============================================================================
 */

#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 10) ---
const float KP_GAIN      = 4.500f;
const float TI_SEC       = 1.500f;
const float TD_SEC       = 0.250f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0500f; // Periodo de muestreo Ts = 50 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 27648.0f;
const float PV_MIN_ENG   = -50.0f;
const float PV_MAX_ENG   = 150.0f;

// --- 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 10"));
  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_10.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
VIDEO 10: Video 10 — Parte 5B: Programación Avanzada de PLC — Temporizadores, Contadores, Registros, MCR, Salto,
Datos y E/S Analógica
Archivo: supervision_sbc_y_sintonizacion_video_10.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 10
KP_NOMINAL = 4.5
TI_NOMINAL = 1.5
TD_NOMINAL = 0.25
N_FILTRO   = 10
TS_SEG     = 0.0500
U_MIN      = 0
U_MAX      = 27648


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 10")
    print(" Enfoque:", "Bloques IEC 61131-3 (TON/TOF/TP/TONR, CTU/CTD/HSC), Shift Registers (SHRB), Zona MCR,
Saltos, Aritmética y Escalado 4-20 mA")
    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=75)
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_10.st)

```
(* ================================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 10: Video 10 — Parte 5B: Programación Avanzada de PLC — Temporizadores, Contadores, Registros, MCR,
Salto, Datos y E/S Analógica
 * Archivo: automatizacion_plc_iec61131_video_10.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_V10) 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_V10
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 := 70.0;
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 := -50.0;
    rMaxIngenieria       : REAL := 150.0;

    (* 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 := 4.500;
    rTi_s                : REAL := 1.500;
    rTd_s                : REAL := 0.250;
```

```
    rTs_s                : REAL := 0.0500;
    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
```