CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA

Video 10 — Parte 5B: Lógica Secuencial, Flip-Flops, Aplicaciones, Circuitos Integrados TTL y CMOS, CI Especiales y Símbolos IEEE — Flip-Flops (Disparo, Entradas Asíncronas, D y JK), Antirrebote, Registros, Contadores, Interfaces, TTL/CMOS, 555, Schmitt, Contador Décadas e IEEE

Parte del Curso: Parte 5B — Lógica Secuencial, Flip-Flops, Aplicaciones, Familias TTL y CMOS, CI Especiales (555, Schmitt, Décadas) y Símbolos IEEE
Arquitectura / Controlador Estudiado: Flip-Flops D y JK (Disparo por Flanco y Preset/Clear), Antirrebote, Registros, Contadores, Interfaz TTL/CMOS, Timer 555 y Norma IEEE Std 91

```
FF-D: Q(t+1) = D | FF-JK: Q(t+1) = J·~Q + ~K·Q | f_out = f_clk / 2^N | 555 Mono: T = 1.1·R·C | 555 Astable: f = 1.44/((R_A+2R_B)C)
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Lógica secuencial y flip-flops: diferencia entre latch por nivel y flip-flop disparado por flanco (setup y hold), entradas asíncronas (Preset y Clear) y funcionamiento de los flip-flops tipo D y JK.
- Aplicaciones fundamentales de los flip-flops: eliminación de rebote de interruptores (switch debouncing), registros de datos paralelos, contadores binarios y divisores de frecuencia, e interfaces serie/paralelo mediante registros de desplazamiento.
- Circuitos integrados TTL y CMOS: interpretación de hojas de datos (V_IH, V_IL, V_OH, V_OL, Fan-Out, potencia dinámica), configuraciones de salida (Totem-Pole/Push-Pull, Open-Collector/Open-Drain y Tri-State) e interconexión segura entre TTL y CMOS (74HCT y resistencias pull-up).
- Circuitos integrados digitales de propósito especial (contador de décadas 74HC90/CD4017, disparador Schmitt 74HC14 y temporizador 555 en modos monoestable y astable) y simbología digital estándar IEEE Std 91 / IEC 60617.

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	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.35 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.05 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	1 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[0, 5]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[0, 5]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Lógica Secuencial y Elementos Biestables Básicos (Latches)

Realimentación Cruzada de Compuertas NAND/NOR, Biestable SR (Set-Reset), Estado de Memoria y Latch D Transparente por Nivel

- **Turno 1 (Masculina):** La lógica secuencial introduce el concepto de estado interno o memoria en los sistemas digitales: la salida actual depende no solo de las entradas presentes, sino también del estado almacenado en instantes anteriores.
- **Turno 2 (Femenina):** El elemento de memoria elemental de un bit es el biestable o latch ese erre, construido conectando dos compuertas NOR o dos compuertas NAND con realimentación cruzada desde la salida de cada una hacia la entrada de la otra.
- **Turno 3 (Masculina):** Un pulso activo en la entrada Set fija la salida cu en uno lógico, un pulso en la entrada Reset la pone en cero lógico, y cuando ambas entradas están inactivas el circuito retiene indefinidamente su último estado almacenado.
- **Turno 4 (Femenina):** Para evitar la condición prohibida donde Set y Reset se activan a la vez, el latch tipo D añade un inversor entre ambas ramas y una entrada de habilitación Enable, siendo transparente mientras dicha habilitación permanece en nivel activo.

Diapositiva 02: Flip-Flops: Disparo por Flanco y Entradas Asíncronas

Diferencia entre Nivel (Latch) y Flanco de Reloj (Flip-Flop Positivo/Negativo), Tiempos de Setup y Hold, y Entradas Preset/Clear

- **Turno 1 (Masculina):** En sistemas síncronos complejos, un latch transparente por nivel resulta problemático porque cualquier cambio en la entrada mientras el reloj está en alto se propaga de inmediato a través de múltiples etapas.
- **Turno 2 (Femenina):** El flip-flop resuelve este problema mediante el disparo por flanco o edge triggering: muestrea sus entradas y actualiza su salida cu única y exclusivamente en el instante preciso de la transición del reloj, ya sea en el flanco de subida o en el de bajada.
- **Turno 3 (Masculina):** En el símbolo gráfico el disparo por flanco se identifica por un pequeño triángulo dinámico en la entrada de reloj, acompañado de un círculo si es sensible al flanco descendente, debiendo respetarse los tiempos de establecimiento setup y de retención hold.
- **Turno 4 (Femenina):** Además de las entradas síncronas que esperan al pulso de reloj, los flip-flops incorporan entradas asíncronas prioritarias llamadas Preset o Set directo y Clear o Reset directo, las cuales fuerzan la salida a uno o a cero de forma inmediata sin esperar al reloj.

Diapositiva 03: El Flip-Flop Tipo D y el Flip-Flop JK

Flip-Flop D (Ecuación Q(t+1) = D, Memoria y Divisor) y Flip-Flop JK Universal (Memoria, Reset, Set y Conmutación Toggle J = K = 1)

- **Turno 1 (Masculina):** Los dos biestables síncronos más utilizados en el diseño digital son el flip-flop tipo D o de datos y el flip-flop jota ka.
- **Turno 2 (Femenina):** El flip-flop tipo D, como el circuito integrado setenta y cuatro HC setenta y cuatro, posee una única entrada síncrona de datos: en cada flanco activo de reloj su ecuación característica establece simplemente que el siguiente estado de la salida cu es igual al valor presente en D.
- **Turno 3 (Masculina):** Por su parte, el flip-flop jota ka es el biestable más versátil: la entrada jota equivale a Set y la entrada ka equivale a Reset, pero elimina la condición ambigua cuando ambas entradas valen uno simultáneamente.
- **Turno 4 (Femenina):** Cuando jota es igual a uno y ka es igual a uno, en cada flanco de reloj la salida cu invierte su estado anterior pasando de cero a uno y de uno a cero, operación denominada modo de conmutación o toggle.

Diapositiva 04: Aplicaciones de los Flip-Flops I: Antirrebote de Interruptores y Registros de Datos

Eliminación de Rebote Mecánico (Switch Debouncing con Latch/Flip-Flop) y Registro Paralelo de N Bits con Reloj Común

- **Turno 1 (Masculina):** La primera aplicación práctica esencial de los biestables es la eliminación del rebote de interruptores o switch debouncing: cuando accionamos un pulsador mecánico, sus láminas metálicas rebotan durante varios milisegundos generando decenas de pulsos falsos.
- **Turno 2 (Femenina):** Conectando un interruptor conmutador de dos posiciones a un latch ese erre, o bien muestreando el pulsador a baja frecuencia mediante un flip-flop tipo D, la salida produce una única transición limpia y libre de rebotes.
- **Turno 3 (Masculina):** La segunda aplicación fundamental es el registro de datos: agrupando cuatro, ocho o dieciséis flip-flops tipo D que comparten una misma línea de reloj y de habilitación formamos un registro paralelo capaz de capturar y retener un byte o palabra completa.
- **Turno 4 (Femenina):** Los registros de datos constituyen la base física de los puertos de salida de los microcontroladores, los registros internos de la unidad aritmético-lógica y los buffers de buses de datos.

Diapositiva 05: Aplicaciones de los Flip-Flops II: Contadores Binarios, Divisores de Frecuencia e Interfaces Serie/Paralelo

División f_out = f_clk / 2^N, Contadores Asíncronos (Ripple) y Síncronos, y Registros de Desplazamiento (SIPO / PISO)

- **Turno 1 (Masculina):** Si en un flip-flop tipo D conectamos su salida negada de regreso a su entrada D, o en un flip-flop jota ka fijamos jota y ka en uno, la salida conmuta en cada pulso y su frecuencia es exactamente la mitad de la frecuencia del reloj de entrada.
- **Turno 2 (Femenina):** Al encadenar ene etapas de este tipo construimos un contador binario de ene bits y a la vez un divisor de frecuencia por potencias de dos, donde la etapa ene divide la frecuencia del reloj entre dos elevado a la ene.
- **Turno 3 (Masculina):** Por otro lado, cuando conectamos la salida cu de cada flip-flop tipo D a la entrada D del flip-flop siguiente con un reloj común, obtenemos un registro de desplazamiento o shift register.
- **Turno 4 (Femenina):** Los registros de desplazamiento realizan la conversión entre interfaces serie y paralelo: convierten tramas serie de un solo cable en palabras paralelas de ocho bits como en el chip setenta y cuatro HC quinientos noventa y cinco, o serializan datos paralelos para transmisión UART y SPI.

Diapositiva 06: Circuitos Integrados TTL y CMOS I: Uso de Hojas de Datos del Fabricante

Familias Lógicas (74LS, 74HC, 74HCT, CD4000), Tensiones V_IH/V_IL/V_OH/V_OL, Corrientes I_OH/I_OL, Fan-Out y Disipación Dinámica

- **Turno 1 (Masculina):** En la implementación de hardware digital coexisten dos grandes tecnologías de circuitos integrados: la familia TTL basada en transistores bipolares, como las series setenta y cuatro LS y ALS, y la familia CMOS basada en transistores MOSFET complementarios, como las series setenta y cuatro HC y cuatro mil.
- **Turno 2 (Femenina):** Al consultar la hoja de datos del fabricante debemos revisar primero los rangos de tensión de alimentación: TTL estándar exige cinco voltios fijos, mientras que CMOS setenta y cuatro HC opera de dos a seis voltios con consumo estático casi nulo.
- **Turno 3 (Masculina):** La hoja de datos especifica las corrientes máximas de entrada y de salida, con las cuales calculamos el factor de carga o fan-out: el número máximo de entradas lógicas que una sola salida puede alimentar de forma confiable sin perder sus niveles válidos.
- **Turno 4 (Femenina):** También debemos observar la capacitancia de disipación de potencia ce pe de, pues aunque una compuerta CMOS casi no consume potencia en reposo, su consumo dinámico crece proporcionalmente a la frecuencia de conmutación y al cuadrado del voltaje de alimentación.

Diapositiva 07: Circuitos Integrados TTL y CMOS II: Configuraciones de Salida de CI Digitales

Salida Totem-Pole / Push-Pull, Salida en Colector/Drenador Abierto (Open-Collector / Open-Drain con Pull-Up) y Salida Tri-Estado (High-Z)

- **Turno 1 (Masculina):** Los circuitos integrados digitales disponen de tres configuraciones internas de etapa de salida que determinan cómo pueden interconectarse entre sí.
- **Turno 2 (Femenina):** La primera es la salida en poste tótem en TTL o push-pull en CMOS, que cuenta con un transistor superior activo hacia ve ce ce y otro inferior hacia tierra; jamás deben unirse dos salidas push-pull directamente entre sí porque si una intenta imponer uno y la otra cero se produce un cortocircuito.
- **Turno 3 (Masculina):** La segunda configuración es la salida de colector abierto o drenador abierto, la cual carece del transistor superior y requiere una resistencia pull-up externa; permite unir múltiples salidas en un mismo cable formando una función AND cableada, como en el bus I dos C.
- **Turno 4 (Femenina):** La tercera configuración es la salida de tres estados o tri-state, que añade un tercer estado de alta impedancia donde ambos transistores se apagan, permitiendo que decenas de memorias y periféricos compartan un mismo bus de datos activando solo uno a la vez.

Diapositiva 08: Circuitos Integrados TTL y CMOS III: Interconexión entre Dispositivos TTL y CMOS

Incompatibilidad Directa TTL -> CMOS (V_OH(min) = 2.7 V vs V_IH(min) = 3.5 V), Solución con 74HCT o Pull-Up, y Conversión 5V <-> 3.3V

- **Turno 1 (Masculina):** Un problema clásico de diseño ocurre al interconectar directamente circuitos integrados de distintas familias o de distintos voltajes de alimentación.
- **Turno 2 (Femenina):** Cuando una salida CMOS de cinco voltios alimenta una entrada TTL setenta y cuatro LS, los niveles de tensión son totalmente compatibles porque CMOS entrega casi cinco voltios en alto y cero voltios en bajo.
- **Turno 3 (Masculina):** Sin embargo, en sentido inverso una salida TTL garantiza apenas dos punto siete voltios como nivel alto mínimo, lo cual resulta insuficiente para una entrada CMOS setenta y cuatro HC que exige al menos tres punto cinco voltios para reconocer un uno seguro.
- **Turno 4 (Femenina):** Para acoplar una salida TTL a una etapa CMOS existen dos soluciones limpias: añadir una resistencia pull-up de cuatro punto siete kiloohmios hacia cinco voltios, o bien emplear la subfamilia intermedia setenta y cuatro HCT cuyas entradas CMOS están diseñadas con umbrales TTL.

Diapositiva 09: Circuitos Integrados Digitales de Propósito Especial: Contador de Décadas, Disparador Schmitt y Temporizador 555

Contador BCD/Décadas (74HC90 / CD4017), Inversor Schmitt Trigger (74HC14) y Temporizador 555 en Modos Monoestable y Astable

- **Turno 1 (Masculina):** Dentro del catálogo digital destacan tres circuitos integrados de propósito especial omnipresentes en proyectos de ingeniería: el contador de décadas, el disparador Schmitt y el temporizador quinientos cincuenta y cinco.
- **Turno 2 (Femenina):** El contador de décadas, como el setenta y cuatro HC noventa en BCD o el CD cuarenta diecisiete con diez salidas decodificadas, cuenta de cero a nueve y se reinicia automáticamente en el décimo pulso, permitiendo dividir frecuencias exactamente por diez o secuenciar etapas.
- **Turno 3 (Masculina):** Las compuertas con entrada disparador Schmitt, como el séxtuple inversor setenta y cuatro HC catorce, incorporan histéresis interna para transformar señales analógicas lentas o ruidosas en ondas cuadradas con flancos perfectamente verticales.
- **Turno 4 (Femenina):** Por último, el legendario temporizador quinientos cincuenta y cinco combina dos comparadores a un tercio y dos tercios de ve ce ce con un flip-flop interno: en modo monoestable genera un pulso preciso de duración uno punto uno por erre por ce, y en modo astable funciona como oscilador de reloj autónomo.

Diapositiva 10: Diseño de Sistemas con Circuitos Integrados y Símbolos Digitales Estándar IEEE

Estándar IEEE Std 91 / IEC 60617-12: Rectángulos Funcionales, Símbolos Calificadores (&, >=1, =1, 1), Bloques SRG/CTR y Notación de Dependencia

- **Turno 1 (Masculina):** Culminamos la Parte cinco con el diseño de sistemas integrados y la lectura de los símbolos lógicos rectangulares normalizados por el estándar IEEE noventa y uno e IEC seiscientos diecisiete.
- **Turno 2 (Femenina):** En los diagramas de hojas de datos internacionales, las compuertas y bloques complejos se representan mediante rectángulos con un símbolo calificador superior: el ampersand identifica la función AND, mayor o igual a uno representa la función OR, e igual a uno identifica la compuerta XOR.
- **Turno 3 (Masculina):** Para circuitos secuenciales de mediana escala, el bloque común de control superior agrupa las líneas de reloj y reset, utilizando el prefijo SRG para registros de desplazamiento, CTR para contadores y MUX para multiplexores.
- **Turno 4 (Femenina):** Su notación de dependencia mediante letras como ce para control, ge para habilitación AND y eme para modo permite comprender de un solo vistazo el funcionamiento interno exacto de cualquier circuito integrado complejo sin necesidad de dibujar cientos de compuertas internas.

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

```
/*
 * ============================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 10: Video 10 — Parte 5B: Lógica Secuencial, Flip-Flops, Aplicaciones, Circuitos Integrados TTL y CMOS, CI Especiales y Símbolos IEEE
 * 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:
 *   Flip-Flops D y JK (Disparo por Flanco y Preset/Clear), Antirrebote, Registros, Contadores, Interfaz TTL/CMOS, Timer 555 y Norma IEEE Std 91
 * ============================================================================
 */

#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      = 4.000f;
const float TI_SEC       = 0.350f;
const float TD_SEC       = 0.050f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0010f; // Periodo de muestreo Ts = 1 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 5.0f;
const float PV_MIN_ENG   = 0.0f;
const float PV_MAX_ENG   = 5.0f;

// --- 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: FUNDAMENTOS DE ELECTRÓNICA
VIDEO 10: Video 10 – Parte 5B: Lógica Secuencial, Flip-Flops, Aplicaciones, Circuitos Integrados TTL y CMOS, CI
Especiales y Símbolos IEEE
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
TI_NOMINAL = 0.35
TD_NOMINAL = 0.05
N_FILTRO   = 10
TS_SEG     = 0.0010
U_MIN      = 0
U_MAX      = 5


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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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


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


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


if __name__ == "__main__":
    print("=" * 78)
    print(" CURSO: FUNDAMENTOS DE ELECTRÓNICA — VIDEO 10")
    print(" Enfoque:", "Flip-Flops D y JK (Disparo por Flanco y Preset/Clear), Antirrebote, Registros, Contadores,
Interfaz TTL/CMOS, Timer 555 y Norma IEEE Std 91")
    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=2.5)
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: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 10: Video 10 — Parte 5B: Lógica Secuencial, Flip-Flops, Aplicaciones, Circuitos Integrados TTL y CMOS, CI
Especiales y Símbolos IEEE
 * 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 := 3.0;
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 := 5.0;

    (* 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 := 0.350;
    rTd_s                : REAL := 0.050;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

END_FUNCTION_BLOCK
```