CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA

Video 01 — Parte 1A: Controladores en Lazo Cerrado, Modos On-Off, Proporcional, Integral, Derivativo y PID — Arquitectura de Realimentación, Histéresis On-Off, Modos P, PI, PD, Estructura PID Estándar y Esquemas Anti-Windup

Parte del Curso: Parte 1A — Controladores en Lazo Cerrado, Modos On-Off, P, I, D y PID
Arquitectura / Controlador Estudiado: Regulador PID Estándar ISA con Derivada sobre la Salida, Filtro Derivativo de 1.er Orden y Anti-Windup

$$u(t) = u0 + Kp \cdot e(t) + (Kp/Ti) \cdot \int e(\tau) d\tau - Kp \cdot Td \cdot (dy_f/dt) + (1/Tt) \cdot \int (u_sat - u) d\tau$$

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Lazo cerrado de realimentación negativa: comparación e(t) = SP - PV, rechazo automático de perturbaciones de carga d(t) y reducción de sensibilidad.
- Control de Dos Posiciones (On-Off) con banda diferencial de histéresis ΔH para evitar el castañeteo mecánico (chattering) de relés y solenoides.
- Modos Proporcional (PB = 100/Kp y origen físico del Offset estacionario), Integral (eliminación exacta del Offset con polo en s = 0) y Derivativo (predicción de tendencia a horizonte Td e inyección de amortiguamiento).
- Prevención del Golpe Derivativo (Derivative Kick) derivando -PV filtrada con divisor N, y esquemas Anti-Windup por Clamping y Back-Calculation.

2. Parámetros Nominales de Configuración del Controlador (Video 01)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	4.2	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.6 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.12 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	2 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[-24, 24]	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: Arquitectura de Control en Lazo Cerrado en Mecatrónica

Planta, Sensor, Comparador, Error Instantáneo e(t) = r(t) - y(t), Actuador y Rechazo de Perturbaciones

- **Turno 1 (Masculina):** Todo sistema mecatrónico de precisión se fundamenta en la arquitectura de control en lazo cerrado: una estructura donde el controlador observa continuamente la salida real de la máquina para corregir cualquier desviación.
- **Turno 2 (Femenina):** El ciclo comienza en el nodo comparador, el cual resta en todo instante la variable medida por el sensor ye de te respecto a la referencia deseada erre de te, generando la señal de error instantáneo e de te.
- **Turno 3 (Masculina):** El controlador procesa matemáticamente ese error y calcula la señal de mando u de te, la cual alimenta al actuador de potencia para modificar el aporte de energía sobre la planta física.
- **Turno 4 (Femenina):** Gracias a esta realimentación negativa, el lazo cerrado compensa de forma automática perturbaciones externas imprevistas, como cambios bruscos de carga mecánica, fluctuaciones de tensión o variaciones térmicas del entorno.

Diapositiva 02: Procesos Continuos frente a Procesos de Eventos Discretos y Híbridos

Regulación de Variables Analógicas frente a Secuenciación Lógica e Integración Mecatrónica

- **Turno 1 (Masculina):** En la industria mecatrónica clasificamos los procesos bajo control en tres grandes categorías según la naturaleza de sus señales y su evolución en el tiempo: procesos continuos, procesos discretos y sistemas híbridos.
- **Turno 2 (Femenina):** En un proceso continuo, tanto las variables medidas como el esfuerzo del actuador toman valores reales ininterrumpidos: por ejemplo, mantener la temperatura de una cámara térmica en ciento ochenta grados o regular el par de un servomotor.
- **Turno 3 (Masculina):** Por el contrario, en un proceso de eventos discretos las variables solo adoptan estados binarios o enteros asociados a sucesos lógicos: presencia de pieza en banda, cilindro extendido, puerta de seguridad cerrada o válvula activada.
- **Turno 4 (Femenina):** Las máquinas mecatrónicas modernas son sistemas intrínsecamente híbridos, donde una máquina de estados o secuencia lógica coordina las etapas operativas mientras lazos continuos regulan simultáneamente la posición, velocidad y fuerza de cada eje.

Diapositiva 03: Terminología Estándar de Control Industrial

Variable de Proceso (PV), Consigna (SP), Variable Manipulada (MV), Offset y Banda Proporcional

- **Turno 1 (Masculina):** Para configurar reguladores industriales, sistemas de control distribuido y bloques de función en ingeniería, es indispensable dominar la nomenclatura internacional estandarizada de instrumentación y control.
- **Turno 2 (Femenina):** Llamamos Variable de Proceso, abreviada pe ve, a la magnitud física real que deseamos gobernar y que es medida por el transductor, mientras que el Setpoint o consigna, abreviado ese pe, es el valor objetivo fijado por el operador o por el planificador.
- **Turno 3 (Masculina):** La Variable Manipulada, abreviada eme ve, es la magnitud sobre la cual actúa directamente el elemento final de control, como el ciclo de trabajo de un puente H o el porcentaje de apertura de una válvula proporcional.
- **Turno 4 (Femenina):** Finalmente, denominamos Offset al error residual permanente que persiste cuando la planta alcanza el régimen estacionario, y Banda Proporcional al porcentaje del rango de medición que provoca el recorrido completo del cien por ciento en la salida del controlador.

Diapositiva 04: Modo de Control de Dos Posiciones (On-Off / Todo o Nada)

Conmutación Discontinua, Zona Muerta Diferencial (Histéresis) y Oscilación en Ciclo Límite

- **Turno 1 (Masculina):** El controlador más sencillo y económico de la ingeniería es el modo de dos posiciones, conocido como control Todo o Nada u On-Off, donde el actuador solo dispone de dos estados extremos: totalmente encendido al cien por ciento o totalmente apagado al cero por ciento.
- **Turno 2 (Femenina):** Si el controlador conmutara exactamente sobre la línea del Setpoint sin ningún margen, el ruido eléctrico del sensor provocaría decenas de conmutaciones por segundo, destruyendo en pocos minutos los contactos de un relé o válvula solenoide.
- **Turno 3 (Masculina):** Para evitar este castañeteo mecánico, introducimos deliberadamente una zona diferencial o banda de histéresis: en un horno, por ejemplo, el calefactor se apaga al superar la consigna más medio grado, y no vuelve a encenderse hasta que la temperatura desciende por debajo de la consigna menos medio grado.
- **Turno 4 (Femenina):** Esta histéresis protege al actuador reduciendo la frecuencia de conmutación, aunque obliga a que la variable de proceso oscile de manera perpetua alrededor de la consigna describiendo un ciclo límite de amplitud controlada.

Diapositiva 05: Modo de Control Proporcional (P) y Banda Proporcional

Ley Lineal u(t) = Kp·e(t) + u0, Relación PB = 100 / Kp y Origen Físico del Error Estacionario (Offset)

- **Turno 1 (Masculina):** Para eliminar las oscilaciones bruscas del modo Todo o Nada, sustituimos el interruptor por el modo de control Proporcional, cuya salida varía de manera continua y linealmente proporcional a la magnitud del error instantáneo.
- **Turno 2 (Femenina):** Su ley matemática establece que la señal de control u de te es igual a la ganancia proporcional ka pe multiplicada por el error e de te, más un término constante de polarización manual u sub cero que representa el esfuerzo nominal cuando el error es nulo.
- **Turno 3 (Masculina):** En instrumentación industrial suele especificarse la Banda Proporcional porcentual, que es igual a cien dividido entre la ganancia ka pe: una ganancia de cuatro equivale a una banda proporcional del veinticinco por ciento.
- **Turno 4 (Femenina):** La limitación fundamental del control puramente proporcional es el Offset: cuando cambia la carga de la planta y se requiere mayor esfuerzo en el actuador que el valor u sub cero, el controlador necesita obligatoriamente que exista un error distinto de cero para sostener esa nueva salida.

Diapositiva 06: Modo de Control Integral (I) y Proporcional-Integral (PI)

Tiempo Integral Ti, Eliminación Matemática del Offset (Polo en el Origen) y Riesgo de Saturación

- **Turno 1 (Masculina):** ¿Cómo logramos que el controlador descubra por sí mismo el esfuerzo exacto que necesita la planta y reduzca el error estacionario exactamente a cero sin intervención humana? Incorporando la acción de control Integral, también llamada de reajuste automático.
- **Turno 2 (Femenina):** En el modo integral, la velocidad de cambio de la señal de control es proporcional al error; dicho de otro modo, la salida acumula la integral del error respecto al tiempo dividida por la constante de tiempo integral te sub i.
- **Turno 3 (Masculina):** Mientras persista el más mínimo error positivo o negativo, la integral seguirá creciendo o decreciendo implacablemente, modificando el mando del actuador hasta que la variable medida coincida con la consigna y el error se anule por completo.
- **Turno 4 (Femenina):** Al combinar la rapidez de respuesta inmediata del término proporcional con la precisión estacionaria del término integral obtenemos el controlador pe i, el más utilizado en la industria, teniendo siempre presente que la acción integral retrasa la fase y puede reducir el margen de estabilidad.

Diapositiva 07: Modo de Control Derivativo (D) y Proporcional-Derivativo (PD)

Tiempo Derivativo Td, Predicción Lineal de Tendencia, Amortiguamiento Activo y Amplificación de Ruido

- **Turno 1 (Masculina):** En sistemas mecatrónicos con alta inercia, como brazos robóticos o carros lineales rápidos, los modos proporcional e integral solo reaccionan al error presente y pasado, llegando demasiado tarde para frenar la masa antes de que sobrepase la meta.
- **Turno 2 (Femenina):** Aquí interviene el modo Derivativo, cuya salida es proporcional a la derivada temporal del error multiplicada por el tiempo derivativo te sub de: no mira cuánto vale el error hoy, sino con qué velocidad y pendiente está cambiando.
- **Turno 3 (Masculina):** Al sumar la acción proporcional con la derivativa en un controlador pe de, el controlador actúa como si predijera el error futuro a una distancia temporal de te sub de segundos, inyectando un amortiguamiento viscoso virtual que frena el sobreimpulso.
- **Turno 4 (Femenina):** Recuerda dos reglas vitales del modo derivativo: primero, nunca puede usarse solo porque ante un error constante su derivada es cero; y segundo, como amplifica el ruido de alta frecuencia del sensor, exige siempre un filtro pasabajos en serie.

Diapositiva 08: Controlador PID Completo y Filtro Derivativo de Primer Orden

Estructuras Paralela Académica frente a Estándar ISA y Eliminación del Golpe Derivativo (Derivative Kick)

- **Turno 1 (Masculina):** Cuando unimos las tres acciones en el controlador pe i de completo, combinamos la fuerza presente del término proporcional, la memoria acumulativa del término integral que elimina el offset, y la anticipación estabilizadora del término derivativo.
- **Turno 2 (Femenina):** En la práctica distinguimos dos formulaciones: la estructura paralela independiente, donde la salida suma ka pe por el error, más ka i por la integral, más ka de por la derivada; y la estructura estándar industrial ISA, donde ka pe multiplica a todo el paréntesis gobernado por los tiempos te sub i y te sub de.
- **Turno 3 (Masculina):** En un controlador industrial real debemos evitar dos fenómenos destructivos en el término derivativo: el primero es el golpe derivativo o Derivative Kick, que ocurre cuando el operador cambia bruscamente el Setpoint en escalón y su derivada tiende a infinito.
- **Turno 4 (Femenina):** Para erradicarlo, calculamos la derivada únicamente sobre menos la variable medida ye de te en lugar de derivar el error; y además acoplamos un filtro derivativo de primer orden con factor divisor ene entre ocho y veinte para bloquear el ruido electrónico.

Diapositiva 09: Saturación del Actuador (Integral Windup) y Esquemas Anti-Windup

Fenómeno de Embalamiento del Integrador, Integración Condicional (Clamping) y Retro-Cálculo (Back-Calculation)

- **Turno 1 (Masculina):** Toda ecuación lineal de control supone que el actuador puede entregar potencia infinita, pero en el mundo físico real un puente H se satura al cien por ciento de ciclo de trabajo, una fuente se limita a veinticuatro voltios y una válvula llega a su tope mecánico.
- **Turno 2 (Femenina):** Cuando el actuador alcanza su saturación física durante un arranque largo, la planta no puede acelerar más rápido; sin embargo, como el error sigue siendo grande, el término integral continúa sumando y acumulando un valor gigantesco en memoria: este defecto se denomina Integral Windup.
- **Turno 3 (Masculina):** La consecuencia del Windup es desastrosa: aunque la planta alcance por fin el Setpoint, el actuador permanece bloqueado al máximo hasta que un error enorme de signo contrario logre descargar toda la integral acumulada, provocando un sobreimpulso violento.
- **Turno 4 (Femenina):** Todo controlador profesional implementa por ello un esquema Anti-Windup: ya sea mediante integración condicional o Clamping, que congela la suma integral en cuanto detecta saturación del mismo signo que el error, o mediante retro-cálculo continuo con constante de seguimiento te sub te.

Diapositiva 10: Síntesis e Integración Práctica de Modos de Control Continuo

Criterios de Selección de Ingeniería (On-Off, P, PI, PD y PID) y Transición al Control Digital

- **Turno 1 (Masculina):** Sinteticemos los criterios de ingeniería de esta Parte uno A para elegir la ley de control óptima frente a cada planta mecatrónica sin sobredimensionar la complejidad del sistema.
- **Turno 2 (Femenina):** Empleamos control On-Off con histéresis en sistemas térmicos o de nivel de gran capacitancia donde una leve oscilación es aceptable; y usamos control pe i en la inmensa mayoría de lazos de velocidad de motores, corriente eléctrica y caudal donde el ruido desaconseja derivar.
- **Turno 3 (Masculina):** Reservamos el control pe de y pe i de completo con filtro derivativo y Anti-Windup para sistemas mecánicos de posicionamiento servo, robótica articular, levitación magnética y control de actitud donde se exige amortiguar oscilaciones rápidas con cero error final.
- **Turno 4 (Femenina):** En el siguiente módulo, correspondiente a la Parte uno B, llevaremos estas leyes continuas al interior de un procesador digital, estudiando su discretización, métricas de desempeño, sintonización de Ziegler-Nichols y control adaptable.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_01.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 01: Video 01 — Parte 1A: Controladores en Lazo Cerrado, Modos On-Off, Proporcional, Integral, Derivativo
y PID
 * Archivo: controlador_tiempo_real_video_01.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:
 *   Regulador PID Estándar ISA con Derivada sobre la Salida, Filtro Derivativo de 1.er Orden y Anti-Windup
 * ============================================================================
 */

#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 01) ---
const float KP_GAIN      = 4.200f;
const float TI_SEC       = 0.600f;
const float TD_SEC       = 0.120f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0020f; // Periodo de muestreo Ts = 2 ms
const float U_MIN_LIM    = -24.0f;
const float U_MAX_LIM    = 24.0f;
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 01"));
  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_01.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
VIDEO 01: Video 01 — Parte 1A: Controladores en Lazo Cerrado, Modos On-Off, Proporcional, Integral, Derivativo y
PID
Archivo: supervision_sbc_y_sintonizacion_video_01.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 01
KP_NOMINAL = 4.2
TI_NOMINAL = 0.6
TD_NOMINAL = 0.12
N_FILTRO   = 10
TS_SEG     = 0.0020
U_MIN      = -24
U_MAX      = 24


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 01")
    print(" Enfoque:", "Regulador PID Estándar ISA con Derivada sobre la Salida, Filtro Derivativo de 1.er Orden y
Anti-Windup")
    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_01.st)

```
(* ================================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 01: Video 01 — Parte 1A: Controladores en Lazo Cerrado, Modos On-Off, Proporcional, Integral, Derivativo
y PID
 * Archivo: automatizacion_plc_iec61131_video_01.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_V01) 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_V01
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 := 4.200;
    rTi_s                : REAL := 0.600;
    rTd_s                : REAL := 0.120;
```

```
    rTs_s                : REAL := 0.0020;
    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
```