CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA

Video 08 — Parte 4B: Amplificador de Instrumentación, Integrador, Derivador, Muestreo y Retención, Comparador y El Op-Amp Real — Amplificador de Instrumentación, Integrador, Diferenciador, Sample and Hold, Comparador y Parámetros de Hojas de Datos del Op-Amp Real

Parte del Curso: Parte 4B — Amplificador de Instrumentación, Integrador, Derivador, Sample & Hold, Comparador y El Op-Amp Real
Arquitectura / Controlador Estudiado: INA de 3 Op-Amps, Integrador/Derivador RC, Sample and Hold, Comparador Schmitt y Parámetros de Hojas de Datos (V_OS, I_B, GBWP, SR, CMRR)

```
INA: A_v = (1 + 2R_1/R_G)·(R_3/R_2) | v_o = -(1/RC)∫v_in dt | v_o = -RC(dv_in/dt) | G_CL·BW = GBWP | f_max = SR/(2π·V_p)
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Amplificador de Instrumentación de 3 Op-Amps (alta impedancia simétrica, ajuste de ganancia con una sola resistencia R_G y rechazo en modo común superior a 100 dB).
- Circuitos de cálculo analógico con Op-Amp: integrador (con resistencia de descarga DC en paralelo) y diferenciador (con resistencia serie de estabilización en alta frecuencia).
- Circuito de Muestreo y Retención (Sample and Hold con interruptor MOSFET y capacitor C_H de baja absorción) y comparadores en lazo abierto y con histéresis (Schmitt Trigger).
- Parámetros reales de las hojas de datos del Op-Amp: tensión de offset V_OS, corrientes de polarización I_B e I_OS, producto ganancia-ancho de banda (GBWP), velocidad de respuesta (Slew Rate), CMRR, PSRR y excursión Rail-to-Rail.

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

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	4.8	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.3 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.05 s, N = 16	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	0.5 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[-12, 12]	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: Amplificador de Instrumentación (INA): Topología de Tres Op-Amps

Etapa de Entrada Diferencial con Buffers Simétricos, Ganancia Ajustable con Una Sola Resistencia R_G y Altísimo CMRR

- **Turno 1 (Masculina):** Para superar todas las limitaciones del restador simple, el amplificador de instrumentación integra tres amplificadores operacionales con resistencias ajustadas por láser dentro de un mismo chip, como el INA ciento veintiocho o el AD seiscientos veinte.
- **Turno 2 (Femenina):** La etapa frontal consta de dos amplificadores no inversores simétricos que reciben directamente las líneas diferenciales ve uno y ve dos, garantizando una impedancia de entrada de gigaohmios en ambos terminales.
- **Turno 3 (Masculina):** Por cortocircuito virtual, la tensión diferencial completa ve dos menos ve uno aparece exactamente sobre la única resistencia externa de ganancia erre sub ge, mientras que la tensión de modo común pasa con ganancia unitaria sin generar corriente en erre sub ge.
- **Turno 4 (Femenina):** De este modo, la ganancia total vale uno más dos erre uno sobre erre sub ge multiplicado por erre tres sobre erre dos, permitiendo fijar ganancias desde uno hasta diez mil cambiando una sola resistencia externa y manteniendo un rechazo en modo común superior a cien decibelios.

Diapositiva 02: Circuito Integrador con Amplificador Operacional

Capacitor C en el Lazo de Realimentación, Ecuación $v_o(t) = -(1/RC)\cdot\int v_{in}(t)dt + v_o(0)$ y Resistencia de Estabilización DC

- **Turno 1 (Masculina):** Si en un amplificador inversor reemplazamos la resistencia de realimentación por un capacitor ce, obtenemos el circuito integrador electrónico, piedra angular del cálculo analógico y del término integral de los controladores PID.
- **Turno 2 (Femenina):** En el nodo de tierra virtual, la corriente de entrada ve in sobre erre debe ser igual a la corriente del capacitor ce por la derivada de menos ve sub o respecto al tiempo.
- **Turno 3 (Masculina):** Integrando ambos miembros encontramos que la tensión de salida ve sub o es igual a menos uno sobre erre por ce multiplicado por la integral en el tiempo de la señal de entrada más la condición inicial del capacitor.
- **Turno 4 (Femenina):** Al aplicar una onda cuadrada de media nula a un integrador obtenemos una onda triangular perfecta; en la práctica se añade una resistencia grande erre efe en paralelo con el capacitor para evitar que pequeñas corrientes de polarización DC saturen el integrador.

Diapositiva 03: Circuito Derivador (Diferenciador) con Amplificador Operacional

Capacitor C en la Entrada y Resistor R en Realimentación, Ecuación $v_o(t) = -R\cdot C\cdot(dv_{in}/dt)$ y Compensación de Ruido de Alta Frecuencia

- **Turno 1 (Masculina):** Intercambiando la posición de los componentes, es decir colocando el capacitor ce en la rama de entrada y la resistencia erre en el lazo de realimentación, construimos el circuito diferenciador o derivador.
- **Turno 2 (Femenina):** Como el nodo inversor está en tierra virtual, la corriente que atraviesa el capacitor de entrada vale ce por la derivada de ve in respecto al tiempo, y al circular por la resistencia erre produce una salida igual a menos erre por ce por la derivada de la entrada.
- **Turno 3 (Masculina):** Este circuito detecta flancos rápidos y convierte una onda triangular en una onda cuadrada, pero presenta un inconveniente inherente: su ganancia en frecuencia crece linealmente con omega, amplificando de forma severa el ruido de alta frecuencia.
- **Turno 4 (Femenina):** Para estabilizar el derivador práctico se coloca siempre una pequeña resistencia erre sub ese en serie con el capacitor de entrada, limitando la ganancia máxima en alta frecuencia y evitando oscilaciones espurias.

Diapositiva 04: Circuito de Muestreo y Retención (Sample and Hold)

Interruptor Analógico MOSFET, Capacitor de Retención C_H de Polipropileno/Poliestireno, Buffers y Conversión Analógico-Digital

- **Turno 1 (Masculina):** Antes de que un conversor analógico-digital de aproximaciones sucesivas pueda cuantificar una señal rápida, necesita congelar su valor instantáneo durante los microsegundos que dura la conversión mediante un circuito de muestreo y retención o sample and hold.
- **Turno 2 (Femenina):** Su estructura consta de un seguidor de tensión de entrada, un interruptor analógico rápido basado en MOSFET, un capacitor de retención ce sub hache y un segundo seguidor de tensión de salida con entrada FET de ultra alta impedancia.
- **Turno 3 (Masculina):** Durante la fase de muestreo o sample, el interruptor se cierra y el buffer de entrada carga rápidamente el capacitor ce sub hache para que siga fielmente a la señal analógica.
- **Turno 4 (Femenina):** Al pasar al modo de retención o hold, el interruptor se abre y el capacitor mantiene constante el voltaje muestreado; se emplean capacitores de polipropileno o teflón de muy baja absorción dieléctrica para minimizar la caída de tensión o droop.

Diapositiva 05: El Comparador Analógico y el Comparador con Histéresis (Schmitt Trigger)

Decisión Binaria de Umbral (v_+ vs v_-), Salida en Colector Abierto (LM393/LM339) y Realimentación Positiva contra Ruido

- **Turno 1 (Masculina):** El comparador es un circuito que tiende un puente directo entre el mundo analógico y el digital: compara continuamente una señal analógica con un voltaje de referencia y entrega una salida lógica de nivel alto o bajo.
- **Turno 2 (Femenina):** Aunque un op-amp en lazo abierto puede funcionar como comparador lento, los comparadores dedicados como el LM trescientos noventa y tres están optimizados para conmutar en nanosegundos sin saturación profunda y suelen tener salida en colector abierto para adaptarse a cualquier voltaje lógico.
- **Turno 3 (Masculina):** Si una señal lenta con ruido superpuesto cruza el umbral de un comparador simple, el ruido provoca múltiples transiciones falsas o rebotes en la salida.
- **Turno 4 (Femenina):** La solución definitiva consiste en aplicar realimentación positiva hacia el terminal no inversor creando un comparador con histéresis o disparador Schmitt, el cual separa el umbral de encendido superior del umbral de apagado inferior.

Diapositiva 06: El Op-Amp Real I: Limitaciones de Excursión, Corriente y Tensión de Offset

Saturación de Rieles vs. Op-Amps Rail-to-Rail, Corriente Máxima de Salida (~25 mA) y Tensión de Offset de Entrada (V_OS)

- **Turno 1 (Masculina):** Para diseñar circuitos profesionales debemos saber interpretar los parámetros no ideales que especifican las hojas de datos o data sheets de los fabricantes para un amplificador operacional real.
- **Turno 2 (Femenina):** El primer límite físico es la excursión de salida: un op-amp clásico como el LM setecientos cuarenta y uno o el TL cero ochenta y dos se satura uno a dos voltios antes de alcanzar sus rieles de alimentación, mientras que un op-amp rail-to-rail moderno alcanza a pocos milivoltios de los rieles.
- **Turno 3 (Masculina):** Además, la etapa de salida interna limita la corriente máxima entregable a unos veinte o cuarenta miliamperios, cayendo la tensión si intentamos conectar cargas de muy baja resistencia.
- **Turno 4 (Femenina):** En corriente continua, debido a leves asimetrías inevitables en el par diferencial de entrada, si unimos ambas entradas a cero voltios aparece una pequeña salida no nula; se modela mediante la tensión de offset de entrada ve o ese, que va de milivoltios en usos generales a microvoltios en op-amps de precisión.

Diapositiva 07: El Op-Amp Real II: Corrientes de Polarización (I_B) y de Offset (I_OS)

Corrientes de Base/Compuerta en las Entradas, Error DC en Resistencias Grandes y Resistencia de Compensación R_comp = R_in || R_f

- **Turno 1 (Masculina):** El segundo grupo de parámetros de corriente continua en la hoja de datos son las corrientes de polarización de entrada i be más e i be menos, que son las pequeñas corrientes requeridas para polarizar los transistores internos de la etapa de entrada.
- **Turno 2 (Femenina):** El promedio de ambas corrientes se denomina corriente de polarización de entrada i sub be: en op-amps con entrada bipolar vale decenas de nanoamperios, mientras que en op-amps con entrada JFET o CMOS baja a unos pocos picoamperios.
- **Turno 3 (Masculina):** Al circular i sub be a través de las resistencias externas del circuito genera caídas de tensión parásitas que se multiplican por la ganancia de la etapa, sumándose al error de offset.
- **Turno 4 (Femenina):** Para cancelar el efecto de la corriente de polarización media conectamos en serie con la entrada no inversora una resistencia de compensación igual al paralelo de erre in con erre efe, quedando únicamente el error residual debido a la corriente de offset i o ese.

Diapositiva 08: El Op-Amp Real III: Respuesta en Frecuencia y Producto Ganancia-Ancho de Banda (GBWP)

Polo Dominante Interno de Compensación, Caída de 20 dB/década, Relación Constante G_CL · BW = f_T y Selección por Velocidad

- **Turno 1 (Masculina):** En corriente alterna, la ganancia en lazo abierto de un amplificador operacional real no es constante con la frecuencia: incorpora un capacitor interno de compensación que introduce un polo dominante a muy baja frecuencia, típicamente entre cinco y cien hercios.
- **Turno 2 (Femenina):** A partir de ese polo dominante, la ganancia en lazo abierto desciende de forma uniforme a razón de menos veinte decibelios por década hasta cruzar la ganancia unitaria de cero decibelios en la frecuencia de transición efe sub te.
- **Turno 3 (Masculina):** Esta caída de primer orden impone un compromiso fundamental en todo op-amp compensado: el producto de la ganancia en lazo cerrado por el ancho de banda resultante es una constante fija llamada Producto Ganancia-Ancho de Banda o GBWP.
- **Turno 4 (Femenina):** Por ejemplo, en un amplificador operacional con producto ganancia-ancho de banda de un megahercio, si configuramos una ganancia de diez obtendremos cien kilohercios de ancho de banda, pero si pedimos una ganancia de cien el ancho de banda se reducirá a solo diez kilohercios.

Diapositiva 09: El Op-Amp Real IV: Slew Rate, CMRR y PSRR en las Hojas de Datos

Tasa Máxima de Cambio SR = |dv_o/dt|_max [V/µs], Ancho de Banda a Plena Potencia f_max = SR/(2π·V_p), CMRR y PSRR en dB

- **Turno 1 (Masculina):** Cuando la señal de salida tiene gran amplitud, entra en juego una limitación no lineal llamada Slew Rate o velocidad de respuesta, medida en voltios por microsegundo, que es la tasa máxima de cambio temporal que puede entregar la salida.
- **Turno 2 (Femenina):** Si exigimos al amplificador una sinusoide cuya pendiente máxima dos pi por efe por el voltaje pico supere el Slew Rate del chip, la salida no podrá seguir a la entrada y se deformará convirtiéndose en una onda triangular.
- **Turno 3 (Masculina):** Asimismo, la hoja de datos especifica el CMRR o relación de rechazo en modo común en decibelios, que mide cuántas veces atenúa el amplificador las tensiones idénticas en ambas entradas frente a la señal diferencial.
- **Turno 4 (Femenina):** Finalmente, el PSRR o relación de rechazo de la fuente de alimentación indica en decibelios la inmunidad del amplificador frente al rizado o ruido presente en sus líneas de alimentación ve ce ce y ve e e.

Diapositiva 10: Síntesis de la Parte 4B: Selección Práctica de Amplificadores Operacionales

Criterios de Ingeniería (Propósito General LM358, Entrada JFET TL082, Precisión OP07, Rail-to-Rail CMOS MCP6002) hacia Sistemas Digitales

- **Turno 1 (Masculina):** Concluimos la Parte cuatro integrando los criterios para elegir el amplificador operacional óptimo según la hoja de datos y la naturaleza del problema.
- **Turno 2 (Femenina):** Para termopares y celdas de carga en continua priorizamos bajo voltaje de offset con chips de precisión como el OP cero siete; para fotodiodos y sensores de alta impedancia elegimos entradas JFET o CMOS de picoamperios como el TL cero ochenta y dos.
- **Turno 3 (Masculina):** Y para sistemas alimentados con batería o microcontroladores de tres punto tres voltios empleamos amplificadores CMOS rail-to-rail de entrada y salida como la familia MCP seis mil dos.
- **Turno 4 (Femenina):** Una vez que la señal analógica ha sido amplificada, filtrada y acondicionada con precisión, estamos listos para entrar en la Parte cinco dedicada al diseño completo de circuitos digitales.

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

```
/*
 * ============================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 08: Video 08 — Parte 4B: Amplificador de Instrumentación, Integrador, Derivador, Muestreo y Retención,
Comparador y El Op-Amp Real
 * Archivo: controlador_tiempo_real_video_08.ino
 * ----------------------------------------------------------------------------
 * Descripción Técnica:
 *   Firmware determinista en C/C++ Embebido para Microcontrolador (compatible
 *   con arquitectura AVR / ARM Cortex-M STM32 / ESP32 / RP2040) que implementa:
 *     1. Acceso a nivel de registros con punteros 'volatile' y máscaras de bits.
 *     2. Controlador PID Discreto por Transformación Bilineal de Tustin con
 *        derivada sobre la variable medida (-PV), filtro derivativo N y
 *        esquema Anti-Windup por retro-cálculo y Clamping.
 *     3. Máquina de Estados Finitos (FSM de Moore) no bloqueante para
 *        secuenciación segura e interbloqueo de emergencia.
 *     4. Trama de telemetría con verificación CRC-16-CCITT hacia SBC/PLC.
 *
 * Enfoque del Módulo:
 *   INA de 3 Op-Amps, Integrador/Derivador RC, Sample and Hold, Comparador Schmitt y Parámetros de Hojas de Datos
(V_OS, I_B, GBWP, SR, CMRR)
 * ============================================================================
 */

#include <Arduino.h>
#include <stdint.h>

// --- Definición de Pines de Interfase (Entradas Optoaisladas y Salidas PWM/MOSFET) ---
const uint8_t PIN_ADC_PV_SENSOR   = A0; // Entrada analógica PV (ej. Shunt 165 ohm de lazo 4-20 mA)
const uint8_t PIN_ADC_SP_CONSIGNA = A1; // Consigna local o referencia analógica (0..5V / 0..3.3V)
const uint8_t PIN_DI_MARCHA_NO    = 2;  // Entrada optoacoplada PC817: Pulsador de Marcha (NO)
const uint8_t PIN_DI_PARO_NC      = 3;  // Entrada optoacoplada PC817: Paro / Emergencia (Cableado NC)
const uint8_t PIN_PWM_ACTUADOR    = 9;  // Salida PWM hacia Driver MOSFET / Puente H
const uint8_t PIN_DO_SSR_ALARMA   = 8;  // Salida hacia Relé de Estado Sólido (SSR) / Piloto

// --- Parámetros Nominales del Controlador PID Discreto (Video 08) ---
const float KP_GAIN      = 4.800f;
const float TI_SEC       = 0.300f;
const float TD_SEC       = 0.050f;
const float N_FILT       = 16.0f;
const float TS_SEC       = 0.0005f; // Periodo de muestreo Ts = 0.5 ms
const float U_MIN_LIM    = -12.0f;
const float U_MAX_LIM    = 12.0f;
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 08"));
  Serial.println(F("# Columnas: t_s,estado_fsm,SP_eng,PV_eng,u_PID,pulsos_enc,crc16"));
  Serial.println(F("# ================================================================"));
}

void loop() {
  uint32_t ahoraMicros = micros();
  uint32_t periodoMicros = (uint32_t)(TS_SEC * 1000000.0f);

  if (ahoraMicros - g_prevMicrosLazo >= periodoMicros) {
    g_prevMicrosLazo += periodoMicros;
    float t_sec = g_pasoMuestreo * TS_SEC;

    // 1. Lectura de Entradas Digitales (Seguridad de Paro NC y Marcha NO)
    bool paroCableadoOK = (digitalRead(PIN_DI_PARO_NC) == HIGH);
    bool ordenMarcha    = (digitalRead(PIN_DI_MARCHA_NO) == LOW);

    // 2. Lectura y Escalado de Entradas Analógicas (NORM_X -> SCALE_X)
    int rawPV = analogRead(PIN_ADC_PV_SENSOR);
    int rawSP = analogRead(PIN_ADC_SP_CONSIGNA);
    float pv_eng = escalarCuentasADC(rawPV, 0, 1023, PV_MIN_ENG, PV_MAX_ENG);
    float sp_eng = escalarCuentasADC(rawSP, 0, 1023, PV_MIN_ENG, PV_MAX_ENG);

    // 3. Evaluación de la Máquina de Estados Finitos (FSM de Moore No Bloqueante)
    if (!paroCableadoOK) {
```

```
    g_estadoActual = ESTADO_FALLA_PARO_NC;
  }

  float mandoActuador = 0.0f;
  switch (g_estadoActual) {
    case ESTADO_REPOSO_SEGURO:
      mandoActuador = 0.0f;
      digitalWrite(PIN_DO_SSR_ALARMA, LOW);
      if (ordenMarcha && paroCableadoOK) {
        g_timestampEstado = millis();
        g_estadoActual    = ESTADO_PRECARGA_SUAVE;
      }
      break;

    case ESTADO_PRECARGA_SUAVE:
      mandoActuador = 0.25f * U_MAX_LIM;
      if (millis() - g_timestampEstado >= 200UL) {
        // Transferencia sin saltos (Bumpless) al activar el lazo PID
        g_pid.integral_acum = mandoActuador;
        g_pid.pv_prev       = pv_eng;
        g_estadoActual      = ESTADO_LAZO_PID_ACTIVO;
      }
      break;

    case ESTADO_LAZO_PID_ACTIVO:
      mandoActuador = evaluarPasoPID_Tustin(g_pid, sp_eng, pv_eng);
      digitalWrite(PIN_DO_SSR_ALARMA, LOW);
      break;

    case ESTADO_FALLA_PARO_NC:
    default:
      mandoActuador = 0.0f;
      g_pid.integral_acum = 0.0f;
      digitalWrite(PIN_DO_SSR_ALARMA, HIGH);
      if (paroCableadoOK && ordenMarcha) {
        g_estadoActual = ESTADO_REPOSO_SEGURO;
      }
      break;
  }

  // 4. Escritura de Salida PWM hacia Etapa de Potencia (MOSFET / Puente H)
  float u_norm_pwm = (mandoActuador - U_MIN_LIM) / (U_MAX_LIM - U_MIN_LIM);
  int duty8bit     = constrain((int)(u_norm_pwm * 255.0f), 0, 255);
  analogWrite(PIN_PWM_ACTUADOR, duty8bit);

  // 5. Lectura Atómica de Contador de Interrupción y Telemetría con CRC-16
  if (g_pasoMuestreo % 10 == 0) {
    noInterrupts();
    uint32_t pulsosCopia = g_contadorPulsosEncoder;
    interrupts();

    uint8_t tramaBytes[6];
    tramaBytes[0] = (uint8_t)g_estadoActual;
    tramaBytes[1] = (uint8_t)duty8bit;
    tramaBytes[2] = (uint8_t)((rawPV >> 8) & 0xFF);
    tramaBytes[3] = (uint8_t)(rawPV & 0xFF);
    tramaBytes[4] = (uint8_t)((pulsosCopia >> 8) & 0xFF);
    tramaBytes[5] = (uint8_t)(pulsosCopia & 0xFF);
    uint16_t crc  = calcularCRC16(tramaBytes, 6);

    Serial.print(t_sec, 3);              Serial.print(',');
    Serial.print((int)g_estadoActual);   Serial.print(',');
    Serial.print(sp_eng, 2);             Serial.print(',');
    Serial.print(pv_eng, 2);             Serial.print(',');
    Serial.print(mandoActuador, 2);      Serial.print(',');
    Serial.print(pulsosCopia);           Serial.print(",0x");
    Serial.println(crc, HEX);
  }
```

```
    g_pasoMuestreo++;
  }
}
```

5. Script Python SBC: Sintonización, Índices IAE/ITAE y Tramas (supervision_sbc_y_sintonizacion_video_08.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
VIDEO 08: Video 08 – Parte 4B: Amplificador de Instrumentación, Integrador, Derivador, Muestreo y Retención,
Comparador y El Op-Amp Real
Archivo: supervision_sbc_y_sintonizacion_video_08.py
--------------------------------------------------------------------------------
Descripción Técnica:
  Script integral en Python para Computadora de Placa Única (Raspberry Pi /
  SBC) y estación de ingeniería que implementa:
    1. Simulación en lazo cerrado de un controlador PID discreto (Tustin +
       Filtro Derivativo N + Anti-Windup por Clamping y Back-Calculation).
    2. Cálculo cuantitativo de métricas transitorias (tr, tp, Mp, ts) y de
       los cuatro índices integrales de desempeño (IAE, ISE, ITAE, ITSE).
    3. Reglas automáticas de sintonización de Ziegler-Nichols (1.º y 2.º
       método), Cohen-Coon y sintonización robusta IMC / Lambda.
    4. Construcción y validación de tramas binarias Maestro-Coprocesador
       (SBC <-> MCU/PLC) protegidas con suma de verificación CRC-16-CCITT.
================================================================================
"""

import math
import struct
from typing import Dict, List, Tuple

# Parámetros nominales del Video 08
KP_NOMINAL = 4.8
TI_NOMINAL = 0.3
TD_NOMINAL = 0.05
N_FILTRO   = 16
TS_SEG     = 0.0005
U_MIN      = -12
U_MAX      = 12


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 08")
    print(" Enfoque:", "INA de 3 Op-Amps, Integrador/Derivador RC, Sample and Hold, Comparador Schmitt y
Parámetros de Hojas de Datos (V_OS, I_B, GBWP, SR, CMRR)")
    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_08.st)

```
(* ================================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 08: Video 08 — Parte 4B: Amplificador de Instrumentación, Integrador, Derivador, Muestreo y Retención,
Comparador y El Op-Amp Real
 * Archivo: automatizacion_plc_iec61131_video_08.st
 * --------------------------------------------------------------------------------
 * Estándar: IEC 61131-3 — Texto Estructurado (ST) + Equivalencia en Lista de
 *           Instrucciones (IL) y Peldaños de Escalera (Ladder Diagram)
 * Descripción Técnica:
 *   Bloque de Función Industrial (FB_ControladorCeldaMecatronica_V08) que
 *   integra el Ciclo de Barrido (Scan), circuito de enclavamiento prioritario
 *   al paro (cableado NC), secuenciador GRAFCET de 4 etapas, temporizador TON,
 *   contador de lotes CTU, registro de corrimiento de rechazo de calidad SHRB,
 *   escalado analógico 4-20 mA (NORM_X / SCALE_X con detección de rotura de
 *   hilo < 3.6 mA) y algoritmo PID discreto con Anti-Windup.
 * ================================================================================ *)

FUNCTION_BLOCK FB_ControladorCeldaMecatronica_V08
VAR_INPUT
    (* Entradas Digitales de la Tabla de Imagen de Proceso PII (%IX) *)
    xPulsadorMarcha_NO   : BOOL; (* %IX0.0 - Contacto Normalmente Abierto *)
    xPulsadorParo_NC     : BOOL; (* %IX0.1 - Cableado Físicamente NC en Campo *)
    xReleTermico_NC      : BOOL; (* %IX0.2 - Guardamotor NC (1 = Sin Falla) *)
    xSensorPiezaEntrada  : BOOL; (* %IX0.3 - Sensor Inductivo PNP 24V DC *)
    xCamaraPiezaDefecto  : BOOL; (* %IX0.4 - 1 = Envase Defectuoso Detectado *)
    xPulsoEncoderBanda   : BOOL; (* %IX0.5 - Reloj de Desplazamiento de Banda *)
    xResetContadores     : BOOL; (* %IX0.6 - Reinicio Manual de Lote *)

    (* Entradas Analógicas Crudas del Convertidor ADC (%IW) *)
    iEntradaAnalogicaRaw : INT;  (* %IW64  - Lazo 4-20 mA: 0 a 27648 cuentas *)
    rSetpointIngenieria  : REAL := 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 := 4.800;
    rTi_s                : REAL := 0.300;
    rTd_s                : REAL := 0.050;
```

```
    rTs_s                : REAL := 0.0005;
    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
```