CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA

Video 02 — Parte 1B: Controladores Digitales, Desempeño, Sintonización, Control de Velocidad y Control Adaptable — Muestreo y ZOH, Discretización por Euler y Tustin, PID Incremental, Índices IAE/ITAE, Ziegler-Nichols y MRAC

Parte del Curso: Parte 1B — Controladores Digitales, Desempeño, Sintonización, Velocidad y Adaptable
Arquitectura / Controlador Estudiado: Algoritmo PID Discreto por Tustin en Forma Incremental (Bumpless), Sintonización Ziegler-Nichols/IMC y MRAC

$$\Delta u[k] = Kp\cdot(e[k] - e[k-1]) + Ki\cdot Ts\cdot(e[k] + e[k-1])/2 - \Delta D_f[k] \mid ITAE = \int t\cdot|e(t)|\,dt$$

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Muestreo periódico Ts y Retenedor de Orden Cero (ZOH con retardo efectivo Ts/2); selección de fs entre 10 y 30 veces el ancho de banda del lazo cerrado.
- Discretización por Euler adelante, Euler atrás y Transformación Bilineal de Tustin s ≈ (2/Ts)·(1 - z^-1)/(1 + z^-1); forma posicional frente a forma incremental de velocidad.
- Métricas temporales (tr, tp, Mp, ts) e índices integrales óptimos (IAE, ISE, ITAE, ITSE); sintonización de Ziegler-Nichols (FOPDT y ganancia última Ku, Pu), Cohen-Coon y reglas IMC Lambda.
- Control de velocidad con encoder en cuadratura (Método M a alta velocidad y Método T a baja velocidad), lazo en cascada corriente-velocidad y control adaptable (Gain Scheduling, MRAC y STR).

2. Parámetros Nominales de Configuración del Controlador (Video 02)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	2.8	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 = 0.06 s, N = 15	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, 255]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[0, 3000]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Arquitectura de un Sistema de Control Digital en Tiempo Discreto

Periodo de Muestreo Ts, Conversión ADC, Retenedor de Orden Cero (ZOH) y Regla de Selección de Frecuencia

- **Turno 1 (Masculina):** En la mecatrónica contemporánea, las leyes de control rara vez se implementan con amplificadores operacionales analógicos; hoy residen como algoritmos numéricos dentro de un microcontrolador, un procesador digital de señales o un PLC.
- **Turno 2 (Femenina):** Un lazo de control digital opera gobernado por una interrupción de reloj periódica cada te sub ese segundos: en cada instante discreto ka, el convertidor analógico-digital toma una muestra de la salida del sensor y la entrega a la unidad central de procesamiento.
- **Turno 3 (Masculina):** El procesador evalúa la ecuación en diferencias en pocos microsegundos y escribe el nuevo comando numérico en el convertidor digital-analógico o modulador pe doble ve eme, el cual mantiene ese valor constante durante todo el intervalo te sub ese: esto modela un Retenedor de Orden Cero.
- **Turno 4 (Femenina):** Dado que el retenedor de orden cero introduce un retardo promedio equivalente a la mitad del periodo de muestreo, la regla de oro de ingeniería exige elegir una frecuencia de muestreo entre diez y treinta veces superior al ancho de banda deseado del lazo cerrado.

Diapositiva 02: Discretización de Controladores Continuos: Euler y Tustin

Aproximación Rectangular hacia Adelante, hacia Atrás y Regla Trapezoidal Bilineal de Tustin

- **Turno 1 (Masculina):** Para traducir un controlador diseñado en el dominio continuo de Laplace al código en C de un microcontrolador, debemos aproximar numéricamente las integrales y las derivadas temporales utilizando muestras discretas separadas por el periodo te sub ese.
- **Turno 2 (Femenina):** El primer método es la aproximación rectangular de Euler hacia adelante, que aproxima el área bajo la curva usando el valor del paso anterior; y el segundo es Euler hacia atrás, que emplea el valor del paso actual y garantiza que cualquier filtro estable continuo siga siendo estable en discreto.
- **Turno 3 (Masculina):** Sin embargo, el estándar de excelencia en ingeniería de control es la regla trapezoidal, conocida en procesamiento de señales como Transformación Bilineal de Tustin.
- **Turno 4 (Femenina):** El método de Tustin calcula la integral como el área del trapecio formado por el promedio entre el error actual e de ka y el error anterior e de ka menos uno multiplicado por te sub ese, logrando la mayor fidelidad en frecuencia sin distorsionar los márgenes de estabilidad.

Diapositiva 03: Algoritmo PID Discreto: Forma Posicional frente a Forma Incremental

Ecuaciones en Diferencias Recursivas, Algoritmo de Velocidad Δu[k] y Transferencia sin Saltos (Bumpless)

- **Turno 1 (Masculina):** Al programar un controlador pe i de discreto en firmware disponemos de dos algoritmos matemáticos equivalentes pero con propiedades operativas muy distintas: la forma posicional y la forma incremental o de velocidad.
- **Turno 2 (Femenina):** En la forma posicional calculamos directamente el mando total u de ka sumando el término proporcional actual, el acumulador integral recursivo y la diferencia filtrada de la variable medida entre el paso actual y el paso previo.
- **Turno 3 (Masculina):** En cambio, en la forma incremental restamos la ecuación del paso ka menos la ecuación del paso ka menos uno, de modo que el algoritmo solo calcula el incremento delta u de ka que debe sumarse a la salida anterior u de ka menos uno.
- **Turno 4 (Femenina):** La forma incremental ofrece dos ventajas extraordinarias en la industria: primero, al acotar directamente la salida u de ka queda protegida de forma natural contra el Windup; y segundo, permite conmutar de modo manual a modo automático sin sacudidas bruscas, logrando una transferencia Bumpless.

Diapositiva 04: Métricas de Desempeño en el Dominio del Tiempo

Tiempo de Subida (tr), Tiempo Pico (tp), Sobrepaso Máximo (Mp), Establecimiento (ts) y Razón de Decaimiento

- **Turno 1 (Masculina):** Para evaluar de manera objetiva si un controlador está bien sintonizado, excitamos el lazo cerrado con un cambio en escalón del Setpoint y medimos cinco especificaciones cuantitativas sobre la curva de respuesta temporal.
- **Turno 2 (Femenina):** La rapidez inicial se mide con el tiempo de subida te sub erre, que es el intervalo que tarda la salida en pasar del diez al noventa por ciento de su cambio final, y con el tiempo pico te sub pe, instante en que alcanza la primera cresta máxima.
- **Turno 3 (Masculina):** La calidad del amortiguamiento se cuantifica mediante el sobrepaso máximo porcentual eme sub pe, que expresa cuánto excede el pico máximo al valor estacionario final, y mediante la razón de decaimiento, cociente entre la altura del segundo pico y el primero.
- **Turno 4 (Femenina):** Por último, el tiempo de establecimiento te sub ese mide el tiempo total requerido para que la variable de proceso entre y permanezca definitivamente dentro de una banda de tolerancia del más menos dos por ciento alrededor de la consigna.

Diapositiva 05: Índices Integrales de Desempeño Óptimo (IAE, ISE, ITAE e ITSE)

Funcionales de Costo para Optimización Computacional de Ganancias y Penalización de Oscilaciones Tardías

- **Turno 1 (Masculina):** Cuando queremos que un algoritmo computacional encuentre automáticamente las mejores ganancias de un controlador, necesitamos condensar toda la curva de error de una maniobra en un único número escalar llamado índice integral de desempeño.
- **Turno 2 (Femenina):** El primer criterio es la Integral del Error Absoluto, abreviada I A E, que mide el área total encerrada entre la referencia y la salida sin importar el signo del error, siendo ideal para minimizar el consumo acumulado de energía o reactivos.
- **Turno 3 (Masculina):** El segundo criterio es la Integral del Error Cuadrático, abreviada I S E: al elevar el error al cuadrado, castiga severamente los errores grandes iniciales que ocurren justo después del escalón, aunque suele producir respuestas agresivas con cierta oscilación posterior.
- **Turno 4 (Femenina):** Por ello, el índice preferido en mecatrónica es la Integral del Error Absoluto Ponderado en el Tiempo, abreviada I T A E: al multiplicar el valor absoluto del error por el tiempo te, tolera el error inevitable del primer instante pero penaliza con dureza cualquier oscilación tardía, logrando asentamientos limpios y rápidos.

Diapositiva 06: Sintonización Empírica de Ziegler-Nichols (1.º y 2.º Método)

Curva de Reacción en Lazo Abierto (Modelo FOPDT: K, L, T) y Ganancia/Periodo Últimos en Lazo Cerrado (Ku, Pu)

- **Turno 1 (Masculina):** En mil novecientos cuarenta y dos, John Ziegler y Nathaniel Nichols publicaron los dos métodos experimentales de sintonización de controladores pe i de más famosos de la historia, diseñados para obtener una razón de decaimiento de un cuarto cuando no se conoce el modelo analítico exacto.
- **Turno 2 (Femenina):** El primer método se realiza en lazo abierto aplicando un escalón en el actuador: sobre la curva de reacción en forma de letra ese trazamos la recta tangente de máxima pendiente para extraer tres parámetros de un modelo de primer orden más tiempo muerto: ganancia estática ka, retardo aparente ele y constante de tiempo te.
- **Turno 3 (Masculina):** Con esos tres números calculamos directamente las ganancias: para un pe i de completo, ka pe vale uno punto dos por te dividido entre ka por ele, el tiempo integral te sub i vale dos veces el retardo ele, y el tiempo derivativo te sub de vale la mitad de ele.
- **Turno 4 (Femenina):** El segundo método se ejecuta en lazo cerrado anulando la acción integral y derivativa: subimos gradualmente la ganancia proporcional hasta que la planta presenta oscilaciones sostenidas con ganancia última ka sub u y periodo último pe sub u, fijando entonces ka pe en cero punto seis por ka sub u, te sub i en la mitad de pe sub u, y te sub de en un octavo de pe sub u.

Diapositiva 07: Sintonización de Cohen-Coon, Control por Modelo Interno (IMC) y Reglas Lambda

Compensación de Procesos con Retardo Dominante y Selección Directa de la Constante de Tiempo en Lazo Cerrado

- **Turno 1 (Masculina):** Aunque las tablas clásicas de Ziegler-Nichols rechazan muy bien las perturbaciones, producen sobreimpulsos cercanos al veinticinco o cuarenta por ciento que resultan inaceptables en robots de ensamblaje o máquinas herramienta de precisión.
- **Turno 2 (Femenina):** Además, cuando el tiempo muerto ele es grande en comparación con la constante de tiempo te, Ziegler-Nichols pierde eficacia; para esos casos el método de Cohen-Coon introduce términos correctivos adicionales basados en la razón de controlabilidad ele sobre te.
- **Turno 3 (Masculina):** Sin embargo, la metodología moderna más elegante y robusta en la industria es el Control por Modelo Interno, conocido como sintonización I M C o método Lambda.
- **Turno 4 (Femenina):** En el método Lambda, el ingeniero elige directamente la constante de tiempo deseada del lazo cerrado lambda sin ningún sobreimpulso: igualando el tiempo integral te sub i a la constante de tiempo de la planta te, y fijando ka pe igual a te dividido entre la ganancia ka multiplicada por lambda más el retardo ele.

Diapositiva 08: Control de Velocidad de Motores en Lazo Cerrado y Lazos en Cascada

Medición con Encoder Cuadratura (Método M frente a Método T), Perfiles S-Curve y Lazo Interno de Corriente

- **Turno 1 (Masculina):** Apliquemos ahora el control digital a uno de los problemas centrales de la mecatrónica: la regulación de velocidad y posición de servomotores mediante encoders incrementales ópticos o magnéticos en cuadratura.
- **Turno 2 (Femenina):** Para estimar digitalmente la velocidad angular existen dos técnicas complementarias: el Método eme cuenta cuántos pulsos de encoder llegan durante una ventana fija de muestreo, ofreciendo gran precisión a alta velocidad pero cuantización ruidosa a baja velocidad.
- **Turno 3 (Masculina):** Por el contrario, el Método te mide con un temporizador de alta frecuencia cuántos pulsos de reloj de megahercios transcurren entre dos flancos consecutivos del encoder, brindando resolución extraordinaria a bajas revoluciones.
- **Turno 4 (Femenina):** En los servodrives industriales combinamos esta medición con perfiles de aceleración suave en curva ese para no excitar vibraciones mecánicas, y estructuramos un control en cascada: un lazo externo de velocidad pe i que entrega la consigna de par a un lazo interno ultrarrápido de corriente de armadura.

Diapositiva 09: Control Adaptable en Mecatrónica: Gain Scheduling, MRAC y STR

Adaptación en Tiempo Real ante Variaciones de Inercia, Carga y No Linealidades Fuertes de la Planta

- **Turno 1 (Masculina):** ¿Qué sucede cuando un brazo robótico extiende su articulación y recoge una carga pesada, haciendo que su momento de inercia cambie cinco veces en un segundo? Un controlador pe i de de ganancias fijas se vuelve lento o entra en oscilación.
- **Turno 2 (Femenina):** Para superar este reto recurrimos al Control Adaptable, cuya primera estrategia es la Programación de Ganancias o Gain Scheduling: medimos una variable auxiliar ligada al punto de operación, como el ángulo articular o la velocidad de vuelo, e interpolamos ganancias precalculadas desde una tabla.
- **Turno 3 (Masculina):** La segunda arquitectura es el Control Adaptable por Modelo de Referencia, abreviado M R A C: ejecutamos en el microcontrolador un modelo matemático ideal que dicta cómo deseamos que responda la planta, y ajustamos las ganancias en cada ciclo para anular el error de seguimiento.
- **Turno 4 (Femenina):** Y la tercera arquitectura es el Regulador Auto-Sintonizable o Self-Tuning Regulator: un estimador de mínimos cuadrados recursivos identifica en línea los parámetros cambiantes de la planta en cada muestra y recalcula al vuelo los coeficientes del controlador digital.

Diapositiva 10: Síntesis de Control Digital, Sintonización y Control Adaptable

Flujo Completo de Diseño del Algoritmo de Control y Transición hacia la Arquitectura de Microcontroladores

- **Turno 1 (Masculina):** Con este módulo completamos la Parte uno del curso, habiendo recorrido el camino completo desde el concepto de realimentación hasta los algoritmos digitales y adaptables más avanzados.
- **Turno 2 (Femenina):** Ahora sabemos elegir el periodo de muestreo te sub ese, discretizar mediante la transformación bilineal de Tustin, programar el pe i de en forma incremental libre de Windup y sintonizarlo con Ziegler-Nichols, reglas Lambda o índices integrales.
- **Turno 3 (Masculina):** Pero todo algoritmo de control digital necesita un cerebro electrónico físico capaz de ejecutar estas ecuaciones en microsegundos con absoluto determinismo temporal.
- **Turno 4 (Femenina):** En la siguiente parte del curso nos adentraremos en el interior del silicio para estudiar la arquitectura de hardware y la programación en tiempo real de microcontroladores y microprocesadores.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_02.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 02: Video 02 — Parte 1B: Controladores Digitales, Desempeño, Sintonización, Control de Velocidad y
Control Adaptable
 * Archivo: controlador_tiempo_real_video_02.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:
 *   Algoritmo PID Discreto por Tustin en Forma Incremental (Bumpless), Sintonización Ziegler-Nichols/IMC y MRAC
 * ============================================================================
 */

#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 02) ---
const float KP_GAIN      = 2.800f;
const float TI_SEC       = 0.350f;
const float TD_SEC       = 0.060f;
const float N_FILT       = 15.0f;
const float TS_SEC       = 0.0010f; // Periodo de muestreo Ts = 1 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 255.0f;
const float PV_MIN_ENG   = 0.0f;
const float PV_MAX_ENG   = 3000.0f;

// --- 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 02"));
  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_02.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
VIDEO 02: Video 02 — Parte 1B: Controladores Digitales, Desempeño, Sintonización, Control de Velocidad y Control
Adaptable
Archivo: supervision_sbc_y_sintonizacion_video_02.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 02
KP_NOMINAL = 2.8
TI_NOMINAL = 0.35
TD_NOMINAL = 0.06
N_FILTRO   = 15
TS_SEG     = 0.0010
U_MIN      = 0
U_MAX      = 255


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 02")
    print(" Enfoque:", "Algoritmo PID Discreto por Tustin en Forma Incremental (Bumpless), Sintonización Ziegler-
Nichols/IMC y MRAC")
    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=1500)
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_02.st)

```
(* ================================================================================
 * CURSO COMPLETO: CONTROLADORES EN MECATRÓNICA
 * VIDEO 02: Video 02 — Parte 1B: Controladores Digitales, Desempeño, Sintonización, Control de Velocidad y
Control Adaptable
 * Archivo: automatizacion_plc_iec61131_video_02.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_V02) 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_V02
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 := 1800.0;
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 := 3000.0;

    (* 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 := 2.800;
    rTi_s                : REAL := 0.350;
    rTd_s                : REAL := 0.060;
```

```
    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
```