CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA

Video 01 — Parte 1A: Elementos Eléctricos Básicos (R, C, L), Leyes de Kirchhoff, Fuentes, Medidores y Circuitos Equivalentes de Thévenin y Norton — Resistor, Capacitor e Inductor, Circuitos en Serie y Paralelo, Fuentes de Tensión y Corriente, Voltímetro, Amperímetro y Teoremas de Redes

Parte del Curso: Parte 1A — Circuitos Eléctricos, Elementos R-L-C, Kirchhoff, Fuentes y Equivalentes de Thévenin y Norton
Arquitectura / Controlador Estudiado: Leyes Constitutivas R, C, L, Nodos y Mallas de Kirchhoff, Divisores, Medidores y Reducción de Redes por Thévenin y Norton

$$v_R = R \cdot i \mid i_C = C \cdot (dv_C/dt) \mid v_L = L \cdot (di_L/dt) \mid V_{th} = I_N \cdot R_{th} \mid v_k = V_s \cdot R_k / R_{eq}$$

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Convención pasiva de signos (p = v·i > 0 absorbe potencia) y leyes constitutivas del resistor (Ley de Ohm y efecto Joule), capacitor (campo eléctrico, continuidad de tensión v_C) e inductor (campo magnético, continuidad de corriente i_L).
- Ley de Corrientes de Kirchhoff (LCK en nodos: Σi_entran = Σi_salen) y Ley de Tensiones de Kirchhoff (LVK en lazos cerrados: Σv = 0).
- Circuitos de resistencias en serie (divisor de tensión) y en paralelo (suma de conductancias y divisor de corriente); fuentes reales de tensión/corriente y conexión correcta del voltímetro (paralelo, alta Z) y amperímetro (serie, baja Z).
- Teorema de Thévenin (V_th en circuito abierto y R_th apagando fuentes independientes), Teorema de Norton (I_N en cortocircuito) y transformación de fuentes.

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)`	3	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.5 s	Elimina el error estacionario (Offset) acumulando la integral del error
Tiempo Derivativo y Filtro Divisor	`Td [s] / N`	Td = 0.06 s, N = 10	Anticipa la tendencia sobre -PV y atenúa el ruido de alta frecuencia
Periodo de Muestreo en Tiempo Real	`Ts [ms]`	1 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[0, 24]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[0, 24]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Introducción a los Circuitos Eléctricos y Variables Fundamentales

Carga Eléctrica, Corriente, Diferencia de Potencial, Potencia Instantánea y Convención Pasiva de Signos

- **Turno 1 (Masculina):** Un circuito eléctrico es una interconexión cerrada de elementos por la cual fluye carga eléctrica para transferir energía o procesar información mediante señales de tensión y corriente.
- **Turno 2 (Femenina):** La corriente eléctrica i de te mide la tasa temporal de flujo de carga en amperios, equivalente a un culombio por segundo, mientras que el voltaje o tensión ve de te cuantifica el trabajo en julios por cada culombio desplazado entre dos nodos.
- **Turno 3 (Masculina):** Para analizar cualquier componente sin ambigüedad aplicamos la convención pasiva de signos: cuando la corriente de referencia entra por la terminal marcada con signo positivo de la tensión, el producto pe igual a ve por i representa potencia absorbida.
- **Turno 4 (Femenina):** Si dicho producto resulta negativo, significa que el elemento está entregando energía neta al resto de la red, como ocurre en una batería o en un generador activo.

Diapositiva 02: Elementos Eléctricos Básicos I: El Resistor y la Ley de Ohm

Relación Lineal v = R·i, Conductancia G, Resistividad Física, Disipación por Efecto Joule y Código de Tolerancias

- **Turno 1 (Masculina):** El primer elemento pasivo fundamental es el resistor, el cual se opone al paso de la corriente eléctrica transformando energía eléctrica de forma irreversible en calor por efecto Joule.
- **Turno 2 (Femenina):** Su comportamiento obedece la ley de Ohm: la caída de tensión ve es directamente proporcional a la corriente i multiplicada por la resistencia erre en ohmios, o bien expresada mediante su recíproco, la conductancia ge en siemens.
- **Turno 3 (Masculina):** Físicamente, la resistencia de un conductor depende de la resistividad del material multiplicada por su longitud y dividida entre el área de su sección transversal, además de variar con la temperatura según su coeficiente térmico.
- **Turno 4 (Femenina):** En el diseño electrónico debemos verificar siempre que la potencia disipada, igual a i al cuadrado por erre o ve al cuadrado sobre erre, permanezca con margen seguro por debajo de la potencia nominal del resistor, ya sea de un cuarto de vatio, medio vatio o superior.

Diapositiva 03: Elementos Eléctricos Básicos II: El Capacitor y el Campo Eléctrico

Capacitancia C = ε·A/d, Ecuación Diferencial i_C = C·(dv_C/dt), Continuidad de Tensión y Energía Almacenada

- **Turno 1 (Masculina):** El capacitor o condensador es un elemento pasivo reactivo capaz de almacenar energía potencial en un campo eléctrico establecido entre dos placas conductoras separadas por un material dieléctrico aislante.
- **Turno 2 (Femenina):** La carga acumulada cu es igual a la capacitancia ce en faradios multiplicada por la tensión ve, por lo que al derivar respecto al tiempo obtenemos su ley constitutiva: la corriente i sub ce es igual a ce por la derivada de la tensión respecto al tiempo.
- **Turno 3 (Masculina):** De esta ecuación se desprenden dos propiedades cruciales: en régimen estacionario de corriente continua, donde la tensión no cambia, el capacitor actúa como un circuito abierto; además, su tensión jamás puede cambiar de forma instantánea.
- **Turno 4 (Femenina):** La energía almacenada en sus placas equivale a un medio de la capacitancia por la tensión al cuadrado, energía que el capacitor devuelve íntegramente al circuito durante los ciclos de descarga sin disipación ideal.

Diapositiva 04: Elementos Eléctricos Básicos III: El Inductor y el Campo Magnético

Ley de Faraday-Lenz, Autoinductancia v_L = L·(di_L/dt), Continuidad de Corriente y Energía Magnética

- **Turno 1 (Masculina):** El tercer elemento pasivo básico es el inductor o bobina, el cual almacena energía en un campo magnético generado cuando una corriente eléctrica circula a través de sus espiras arrolladas sobre un núcleo.
- **Turno 2 (Femenina):** Por la ley de inducción de Faraday, cualquier variación temporal de la corriente produce un flujo magnético variable que induce una tensión ve sub ele igual a la inductancia ele en henrios por la derivada de la corriente respecto al tiempo.
- **Turno 3 (Masculina):** De forma dual al capacitor, en régimen permanente de corriente continua la derivada de la corriente es nula y el inductor ideal se comporta como un cortocircuito perfecto, oponiéndose únicamente a los cambios bruscos de corriente.
- **Turno 4 (Femenina):** Como la corriente en un inductor no puede saltar instantáneamente, interrumpir de golpe el paso de corriente genera un pico de alta tensión inductiva, almacenando una energía magnética igual a un medio de ele por la corriente al cuadrado.

Diapositiva 05: Leyes de Kirchhoff y Circuito de Resistencias en Serie

Ley de Corrientes en Nodos (LCK), Ley de Tensiones en Mallas (LVK), Resistencia Equivalente y Divisor de Tensión

- **Turno 1 (Masculina):** Toda red eléctrica concentrada obedece las dos leyes de Kirchhoff: la Ley de Corrientes de Kirchhoff establece por conservación de carga que la suma algebraica de las corrientes que entran y salen de cualquier nodo es exactamente igual a cero.
- **Turno 2 (Femenina):** Por su parte, la Ley de Tensiones de Kirchhoff expresa la conservación de la energía: la suma algebraica de todas las subidas y caídas de potencial alrededor de cualquier trayectoria cerrada o malla es idénticamente cero.
- **Turno 3 (Masculina):** Cuando conectamos varios resistores en serie, todos comparten exactamente la misma corriente y la resistencia equivalente total es la suma directa de cada resistencia individual.
- **Turno 4 (Femenina):** Aplicando la ley de tensiones al circuito serie obtenemos la regla del divisor de tensión: el voltaje en un resistor específico es igual a la tensión total de la fuente multiplicada por dicho resistor y dividida entre la suma de las resistencias en serie.

Diapositiva 06: Circuito de Resistencias en Paralelo y Divisor de Corriente

Nodos Comunes, Suma de Conductancias G_eq = ΣG_k, Fórmula de Dos Ramas y Reparto Inverso de Corriente

- **Turno 1 (Masculina):** Dos o más elementos están conectados en paralelo cuando sus terminales comparten exactamente el mismo par de nodos, quedando sometidos a idéntica diferencia de potencial.
- **Turno 2 (Femenina):** Al aplicar la Ley de Corrientes de Kirchhoff en el nodo superior, la corriente total es la suma de las corrientes de cada rama, por lo que el recíproco de la resistencia equivalente es la suma de los recíprocos de cada resistencia.
- **Turno 3 (Masculina):** En el caso frecuente de dos resistores erre uno y erre dos en paralelo, la resistencia equivalente se calcula rápidamente como el producto de ambas dividido entre su suma, resultando siempre menor que la menor de las resistencias conectadas.
- **Turno 4 (Femenina):** Asimismo, la fórmula del divisor de corriente indica que la corriente que atraviesa una rama es proporcional a la resistencia de la rama opuesta dividida entre la suma de ambas resistencias, concentrándose mayor corriente por el camino de menor oposición.

Diapositiva 07: Fuentes de Tensión y Corriente, y Medidores Eléctricos

Fuentes Ideales frente a Fuentes Reales con Resistencia Interna, Voltímetro en Paralelo y Amperímetro en Serie

- **Turno 1 (Masculina):** Una fuente ideal de tensión mantiene un voltaje fijo en sus terminales independientemente de la corriente demandada, mientras que una fuente real presenta una resistencia interna en serie que provoca caída de tensión al aumentar la carga.
- **Turno 2 (Femenina):** De manera dual, una fuente ideal de corriente suministra una intensidad constante sin importar el voltaje en sus bornes, mientras que una fuente real se modela con una resistencia interna conectada en paralelo.
- **Turno 3 (Masculina):** Para medir una diferencia de potencial conectamos el voltímetro siempre en paralelo con el elemento; un voltímetro ideal posee resistencia interna infinita, típicamente diez megaohmios en multímetros digitales, para no desviar corriente del circuito.
- **Turno 4 (Femenina):** En cambio, el amperímetro debe intercalarse abriendo la rama para quedar en serie con la carga, y debe poseer una resistencia interna casi nula para no introducir una caída de tensión parásita en la medición.

Diapositiva 08: Circuitos Equivalentes de Thévenin: Reducción de Redes Lineales

Tensión de Circuito Abierto V_th, Resistencia Equivalente R_th Apagando Fuentes Independientes y Cálculo de Carga

- **Turno 1 (Masculina):** El teorema de Thévenin demuestra que cualquier red lineal bilateral de fuentes y resistencias vista desde dos terminales A y B puede reemplazarse por una única fuente ideal de tensión ve de Thévenin en serie con una resistencia erre de Thévenin.
- **Turno 2 (Femenina):** El voltaje de Thévenin es exactamente la tensión medida en circuito abierto entre los terminales A y B cuando desconectamos la resistencia de carga externa.
- **Turno 3 (Masculina):** Para hallar la resistencia de Thévenin apagamos todas las fuentes independientes de la red: las fuentes de tensión se reemplazan por cortocircuitos y las fuentes de corriente se reemplazan por circuitos abiertos, calculando la resistencia vista desde los terminales.
- **Turno 4 (Femenina):** Este equivalente simplifica enormemente el diseño electrónico, pues permite evaluar al instante cómo cambiarán el voltaje y la corriente ante distintos valores de carga sin recalcular toda la red original.

Diapositiva 09: Circuito Equivalente de Norton y Transformación de Fuentes

Corriente de Cortocircuito I_N = V_th / R_th, Dualidad Serie-Paralelo y Equivalencia Externa

- **Turno 1 (Masculina):** El teorema de Norton constituye el dual exacto de Thévenin: permite representar la misma red lineal mediante una fuente de corriente i de Norton conectada en paralelo con una resistencia equivalente erre de Norton.
- **Turno 2 (Femenina):** La corriente de Norton se determina colocando un cortocircuito directo entre los terminales de salida A y B y calculando la corriente de cortocircuito resultante, siendo la resistencia de Norton idéntica a la resistencia de Thévenin.
- **Turno 3 (Masculina):** Ambos modelos se relacionan directamente por la ley de Ohm: el voltaje de Thévenin es igual al producto de la corriente de Norton por la resistencia de Thévenin, propiedad conocida como transformación de fuentes.
- **Turno 4 (Femenina):** En el análisis de circuitos complejos podemos alternar sucesivamente entre representaciones de Thévenin en serie y de Norton en paralelo para reducir mallas múltiples a un solo lazo elemental.

Diapositiva 10: Síntesis de la Parte 1A y Conexión con Corriente Alterna y Laboratorio

Resumen de Leyes Constitutivas RLC, Kirchhoff y Equivalentes Lineales hacia el Régimen Senoidal

- **Turno 1 (Masculina):** En esta primera mitad de la Parte uno hemos establecido los cimientos del análisis de circuitos: las leyes constitutivas de resistores, capacitores e inductores junto con las leyes de nodos y mallas de Kirchhoff.
- **Turno 2 (Femenina):** Comprendimos cómo los divisores de tensión y corriente, sumados a los equivalentes de Thévenin y Norton, permiten modelar la interacción real entre fuentes, sensores, etapas amplificadoras y medidores.
- **Turno 3 (Masculina):** Recuerda que la resistencia interna de la fuente y la impedancia de entrada del instrumento determinan el error de carga en cualquier medición electrónica.
- **Turno 4 (Femenina):** En la Parte uno B extenderemos estas leyes al régimen de corriente alterna mediante fasores e impedancias, estudiaremos potencia, transformadores y abordaremos las consideraciones prácticas de prototipado, soldadura, osciloscopio y seguridad eléctrica.

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

```
/*
 * ============================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 01: Video 01 — Parte 1A: Elementos Eléctricos Básicos (R, C, L), Leyes de Kirchhoff, Fuentes, Medidores y
Circuitos Equivalentes de Thévenin y Norton
 * 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:
 *   Leyes Constitutivas R, C, L, Nodos y Mallas de Kirchhoff, Divisores, Medidores y Reducción de Redes por
Thévenin y Norton
 * ============================================================================
 */

#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      = 3.000f;
const float TI_SEC       = 0.500f;
const float TD_SEC       = 0.060f;
const float N_FILT       = 10.0f;
const float TS_SEC       = 0.0010f; // Periodo de muestreo Ts = 1 ms
const float U_MIN_LIM    = 0.0f;
const float U_MAX_LIM    = 24.0f;
const float PV_MIN_ENG   = 0.0f;
const float PV_MAX_ENG   = 24.0f;

// --- 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: FUNDAMENTOS DE ELECTRÓNICA
VIDEO 01: Video 01 – Parte 1A: Elementos Eléctricos Básicos (R, C, L), Leyes de Kirchhoff, Fuentes, Medidores y
Circuitos Equivalentes de Thévenin y Norton
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 = 3
TI_NOMINAL = 0.5
TD_NOMINAL = 0.06
N_FILTRO   = 10
TS_SEG     = 0.0010
U_MIN      = 0
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: FUNDAMENTOS DE ELECTRÓNICA — VIDEO 01")
    print(" Enfoque:", "Leyes Constitutivas R, C, L, Nodos y Mallas de Kirchhoff, Divisores, Medidores y Reducción
de Redes por Thévenin y Norton")
    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=12)
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: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 01: Video 01 — Parte 1A: Elementos Eléctricos Básicos (R, C, L), Leyes de Kirchhoff, Fuentes, Medidores y
Circuitos Equivalentes de Thévenin y Norton
 * 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 := 14.4;
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 := 24.0;

    (* 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 := 3.000;
    rTi_s                : REAL := 0.500;
    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
```