CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA

Video 04 — Parte 2B: Transistor Bipolar (BJT), Emisor Común, Interruptor, Encapsulados, Darlington, Fototransistor y Transistores FET/MOSFET — Física del BJT, Emisor Común, BJT Switch, Encapsulados, Darlington, Optoacoplador y Comportamiento, Símbolos y Aplicaciones de MOSFETs

Parte del Curso: Parte 2B — Transistores Bipolares (BJT), Emisor Común, Interruptor, Darlington, Optoaislador y Transistores FET/MOSFET
Arquitectura / Controlador Estudiado: Física y Regiones del BJT, Amplificador Emisor Común, BJT Switch, Encapsulados, Darlington, Optoacoplador y Conmutación MOSFET

$$I_C = \beta \cdot I_B \mid I_E = I_C + I_B \mid A_v \approx -R_C / R_E \mid \beta_{Darlington} \approx \beta_1 \cdot \beta_2 \mid I_D = K_n \cdot (V_{GS} - V_{GS(th)})^2$$

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Física del transistor bipolar BJT (NPN y PNP) y operación en corte, región activa lineal (amplificación $I_C = \beta \cdot I_B$) y saturación profunda ($V_{CE(sat)} \approx 0.2$ V).
- Circuito de emisor común (recta de carga, punto Q estable con divisor de tensión y resistencia de emisor) y diseño del BJT como interruptor forzado en saturación.
- Encapsulados y resistencia térmica (TO-92, SOT-23, TO-220), conexión Darlington de altísima ganancia (ULN2803) y aislamiento galvánico óptico con fototransistores y optoaisladores (CTR).
- Transistores de efecto de campo (JFET y MOSFET de empobrecimiento e incremento, canal N y canal P): control por campo eléctrico con impedancia de entrada de teraohmios, símbolos normalizados y aplicaciones de MOSFETs en potencia (baja $R_{DS(on)}$, puentes H y lógica CMOS).

2. Parámetros Nominales de Configuración del Controlador (Video 04)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	4.5	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 = 0.04 s, N = 12	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]`	[0, 24]	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: Transistor de Unión Bipolar (BJT): Física y Estructura Interna

Arquitecturas NPN y PNP, Emisor Fuertemente Dopado, Base Muy Delgada, Colector y Ganancia de Corriente β (h_FE)

- **Turno 1 (Masculina):** El transistor de unión bipolar o BJT es un dispositivo semiconductor de tres regiones dopadas alternadas, ya sea en estructura ene pe ene o pe ene pe, que dispone de tres terminales: emisor, base y colector.
- **Turno 2 (Femenina):** Su física operativa depende de una geometría asimétrica deliberada: el emisor está fuertemente dopado para inyectar portadores, mientras que la región central de base es extremadamente delgada y ligeramente dopada.
- **Turno 3 (Masculina):** En un transistor ene pe ene polarizado con la unión base-emisor en directa y la unión colector-base en inversa, más del noventa y nueve por ciento de los electrones inyectados por el emisor atraviesan la base sin recombinarse y son barridos hacia el colector.
- **Turno 4 (Femenina):** De este modo, una diminuta corriente de control en la base gobierna una corriente de colector beta veces mayor, donde beta o hache efe e suele situarse entre cincuenta y trescientos.

Diapositiva 02: Regiones de Operación y Curvas Características del BJT

Corte (I_B = 0), Región Activa Lineal (I_C = β·I_B para Amplificación) y Saturación (V_CE(sat) ≈ 0.2 V para Conmutación)

- **Turno 1 (Masculina):** Por la Ley de Corrientes de Kirchhoff en el transistor, la corriente de emisor es siempre igual a la suma de la corriente de colector más la pequeña corriente de base.
- **Turno 2 (Femenina):** El transistor bipolar puede operar en tres regiones bien diferenciadas sobre sus curvas de salida: cuando la tensión base-emisor es menor a cero punto siete voltios, la corriente de base es nula y el dispositivo está en corte como un interruptor abierto.
- **Turno 3 (Masculina):** En la región activa lineal, la unión base-emisor conduce a unos cero punto siete voltios y la tensión colector-emisor supera el voltio, comportándose el colector como una fuente de corriente controlada igual a beta por la corriente de base.
- **Turno 4 (Femenina):** Finalmente, en la región de saturación ambas uniones quedan en directa, la tensión colector-emisor cae a unos cero punto dos voltios y la corriente máxima queda limitada únicamente por la carga externa, actuando como un interruptor cerrado.

Diapositiva 03: Circuito de Emisor Común: Polarización, Punto Q y Amplificación

Recta de Carga Estática, Polarización Estable por Divisor de Tensión con Resistencia de Emisor y Ganancia de Tensión

- **Turno 1 (Masculina):** La configuración más versátil para amplificación lineal es el circuito de emisor común, donde la señal de entrada se aplica a la base, la salida se toma del colector y el emisor sirve como referencia común.
- **Turno 2 (Femenina):** Al aplicar la Ley de Tensiones de Kirchhoff en la malla de salida trazamos la recta de carga estática: la tensión colector-emisor es igual a la alimentación ve ce ce menos el producto de la corriente de colector por la resistencia de colector.
- **Turno 3 (Masculina):** Para que el punto de operación en reposo cu sea inmune a las grandes variaciones de beta entre transistores y con la temperatura, empleamos un divisor de tensión en la base combinado con una resistencia degenerativa en el emisor.
- **Turno 4 (Femenina):** Cuando superponemos una pequeña señal alterna a través de un capacitor de acoplamiento, el circuito amplifica e invierte la fase con una ganancia de tensión igual a menos la resistencia de colector dividida entre la resistencia de emisor.

Diapositiva 04: El Transistor Bipolar como Interruptor (BJT Switch)

Diseño en Corte y Saturación Profunda, Factor de Sobresaturación, Resistencia de Base R_B y Excitación de Cargas

- **Turno 1 (Masculina):** En sistemas digitales y mecatrónicos utilizamos constantemente el transistor bipolar como interruptor electrónico para activar relés, solenoides, lámparas o indicadores desde un pin lógico de baja corriente.
- **Turno 2 (Femenina):** Al operar como interruptor evitamos por completo la región activa lineal, pues allí el producto de tensión colector-emisor por corriente de colector es elevado y calentaría excesivamente el encapsulado.
- **Turno 3 (Masculina):** Para garantizar una saturación franca con caída de apenas cero punto dos voltios, calculamos la corriente mínima de base dividiendo la corriente de carga entre el beta mínimo y la multiplicamos por un factor de seguridad de dos a cinco.
- **Turno 4 (Femenina):** Con esa corriente de base forzada dimensionamos la resistencia de base erre sub be restando cero punto siete voltios al voltaje de salida del microcontrolador y dividiendo entre la corriente de base deseada.

Diapositiva 05: Encapsulados de Transistores Bipolares y Gestión Térmica

Encapsulados de Señal (TO-92, SOT-23) y de Potencia (TO-220, TO-247, DPAK), Disipadores y Resistencia Térmica θ_JA

- **Turno 1 (Masculina):** La capacidad de corriente y potencia de un transistor depende directamente de su encapsulado físico y de su habilidad para evacuar el calor generado en la unión semiconductora hacia el ambiente.
- **Turno 2 (Femenina):** Los transistores de pequeña señal como el dos ene veintidós veintidós o el BC quinientos cuarenta y siete vienen en encapsulados plásticos TO noventa y dos o superficiales SOT veintitrés, disipando entre doscientos y seiscientos milivatios.
- **Turno 3 (Masculina):** Para corrientes de varios amperios empleamos encapsulados de potencia como el TO doscientos veinte, el DPAK o el TO doscientos cuarenta y siete, los cuales incorporan una aleta metálica conectada internamente al colector para fijarse a un disipador de aluminio.
- **Turno 4 (Femenina):** Utilizando la analogía térmica de la ley de Ohm, el aumento de temperatura de la unión es igual a la potencia disipada multiplicada por la suma de las resistencias térmicas unión-cápsula, cápsula-disipador y disipador-ambiente.

Diapositiva 06: Transistor Darlington: Superganancia de Corriente en Cascada

Conexión Compuesta de Dos BJTs, Ganancia Total β_D = β_1·β_2, Caída Base-Emisor de 1.4 V y Arreglos Integrados ULN2803

- **Turno 1 (Masculina):** Cuando un único transistor bipolar no ofrece suficiente ganancia de corriente para mover una carga pesada a partir de una señal de pocos microamperios, recurrimos a la configuración Darlington.
- **Turno 2 (Femenina):** El par Darlington conecta dos transistores bipolares en cascada dentro de un mismo encapsulado: el emisor del primer transistor piloto alimenta directamente la base del segundo transistor de potencia, mientras ambos colectores se unen.
- **Turno 3 (Masculina):** Gracias a esta topología, la ganancia de corriente total es aproximadamente el producto de ambas ganancias individuales, superando fácilmente valores de mil a diez mil.
- **Turno 4 (Femenina):** Como contrapartida, el par Darlington presenta el doble de caída base-emisor, alrededor de uno punto cuatro voltios, y una tensión de saturación colector-emisor cercana a cero punto nueve voltios, siendo muy popular en circuitos integrados de ocho canales como el ULN veintiocho cero tres.

Diapositiva 07: Fototransistor y Optoaislador: Aislamiento Galvánico Óptico

Base Fotosensible, Relación de Transferencia de Corriente (CTR), Rechazo de Ruido Industrial y Barrera de Kilovoltios

- **Turno 1 (Masculina):** El fototransistor es un transistor bipolar cuya unión colector-base posee una ventana óptica sensible a la luz: los fotones incidentes generan la corriente de base y el transistor la multiplica por su ganancia beta.
- **Turno 2 (Femenina):** Al encapsular un diodo emisor infrarrojo frente a un fototransistor dentro de un cuerpo opaco obtenemos el optoaislador u optoacoplador, como el clásico PC ochocientos diecisiete o el cuatro ene treinta y cinco.
- **Turno 3 (Masculina):** El parámetro clave de su hoja de datos es el CTR o relación de transferencia de corriente, definido como el porcentaje de corriente de colector obtenida en la salida respecto a la corriente inyectada en el LED de entrada.
- **Turno 4 (Femenina):** El optoaislador transmite señales digitales o analógicas mediante luz sin ninguna conexión eléctrica física, soportando barreras de aislamiento de tres mil a cinco mil voltios entre la etapa lógica y la planta de potencia.

Diapositiva 08: Transistores de Efecto de Campo (FET): Comportamiento Físico

Dispositivo Unipolar Controlado por Tensión (Terminales Gate, Drain y Source), JFET frente a MOSFET y Compuerta Aislada

- **Turno 1 (Masculina):** A diferencia del transistor bipolar que es controlado por corriente de base, el transistor de efecto de campo o FET es un dispositivo unipolar donde la corriente entre drenador y fuente es controlada por el campo eléctrico de la terminal de compuerta o gate.
- **Turno 2 (Femenina):** En un JFET o transistor de efecto de campo de unión, la compuerta es una unión pe ene polarizada en inversa cuyo ancho de empobrecimiento estrangula el canal conductor.
- **Turno 3 (Masculina):** Por su parte, en el MOSFET la compuerta metálica o de polisilicio está físicamente separada del canal semiconductor por una finísima capa aislante de dióxido de silicio.
- **Turno 4 (Femenina):** Esta capa de óxido confiere al MOSFET una impedancia de entrada prácticamente infinita en corriente continua, del orden de teraohmios, exigiendo corriente de compuerta casi nula en estado estacionario.

Diapositiva 09: Símbolos que Representan a los Transistores de Efecto de Campo

Identificación Gráfica de JFET y MOSFET, Canal N frente a Canal P (Dirección de la Flecha) y Modo Empobrecimiento vs. Incremento

- **Turno 1 (Masculina):** En la lectura de planos electrónicos es indispensable distinguir con precisión los símbolos normalizados de las distintas familias de transistores de efecto de campo.
- **Turno 2 (Femenina):** En todos los símbolos FET, el sentido de la flecha en el sustrato o en la fuente identifica el tipo de portador: una flecha que apunta hacia adentro del canal indica canal ene, mientras que una flecha que apunta hacia afuera indica canal pe.
- **Turno 3 (Masculina):** En los MOSFET de modo empobrecimiento o depletion, donde el canal existe físicamente con tensión compuerta-fuente igual a cero, la línea vertical entre drenador y fuente se dibuja con trazo continuo.
- **Turno 4 (Femenina):** En cambio, en los MOSFET de modo incremento o enhancement, que son los más utilizados en electrónica de potencia y digital, la línea vertical del canal se dibuja segmentada en tres trazos para indicar que el canal está normalmente abierto hasta superar la tensión umbral ve ge ese th.

Diapositiva 10: Aplicaciones de los MOSFETs en Potencia, Conmutación y Lógica

Región Óhmica de Baja R_DS(on), Conmutación PWM de Motores, Puentes H, Conversores Conmutados y Compuertas CMOS

- **Turno 1 (Masculina):** El MOSFET de incremento domina la electrónica moderna gracias a que en conducción plena actúa como una resistencia óhmica pura erre de ese on de apenas unos pocos miliohmios, reduciendo drásticamente las pérdidas por calor.
- **Turno 2 (Femenina):** Al usar un MOSFET canal ene como interruptor de lado bajo con un microcontrolador de tres punto tres o cinco voltios, debemos elegir un MOSFET de nivel lógico cuya hoja de datos garantice baja resistencia erre de ese on a ese voltaje de compuerta.
- **Turno 3 (Masculina):** A frecuencias de conmutación elevadas debemos recordar que la compuerta presenta una capacitancia parásita de entrada que requiere picos breves de corriente mediante un driver de compuerta para cargar y descargar rápidamente el canal.
- **Turno 4 (Femenina):** Además de gobernar motores en puentes H y fuentes conmutadas, la combinación complementaria de un MOSFET canal pe y un MOSFET canal ene forma el inversor CMOS, base física de todos los microprocesadores y circuitos digitales actuales.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_04.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 04: Video 04 — Parte 2B: Transistor Bipolar (BJT), Emisor Común, Interruptor, Encapsulados, Darlington,
Fototransistor y Transistores FET/MOSFET
 * Archivo: controlador_tiempo_real_video_04.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:
 *   Física y Regiones del BJT, Amplificador Emisor Común, BJT Switch, Encapsulados, Darlington, Optoacoplador y
Conmutación MOSFET
 * ============================================================================
 */

#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 04) ---
const float KP_GAIN      = 4.500f;
const float TI_SEC       = 0.300f;
const float TD_SEC       = 0.040f;
const float N_FILT       = 12.0f;
const float TS_SEC       = 0.0005f; // Periodo de muestreo Ts = 0.5 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   = 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 04"));
  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_04.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
VIDEO 04: Video 04 — Parte 2B: Transistor Bipolar (BJT), Emisor Común, Interruptor, Encapsulados, Darlington,
Fototransistor y Transistores FET/MOSFET
Archivo: supervision_sbc_y_sintonizacion_video_04.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 04
KP_NOMINAL = 4.5
TI_NOMINAL = 0.3
TD_NOMINAL = 0.04
N_FILTRO   = 12
TS_SEG     = 0.0005
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 04")
    print(" Enfoque:", "Física y Regiones del BJT, Amplificador Emisor Común, BJT Switch, Encapsulados,
Darlington, Optoacoplador y Conmutación MOSFET")
    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_04.st)

```
(* ==========================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 04: Video 04 — Parte 2B: Transistor Bipolar (BJT), Emisor Común, Interruptor, Encapsulados, Darlington,
Fototransistor y Transistores FET/MOSFET
 * Archivo: automatizacion_plc_iec61131_video_04.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_V04) 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_V04
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 := 4.500;
    rTi_s                : REAL := 0.300;
    rTd_s                : REAL := 0.040;
```

```
    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
```