CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA

Video 05 — Parte 3A: Respuesta del Sistema, Linealidad de Amplitud, Series de Fourier, Ancho de Banda, Linealidad de Fase y Distorsión — Superposición, Representación de Fourier, Respuesta en Frecuencia y Ancho de Banda, Fase Lineal, Retardo de Grupo y Distorsión de Señales

Parte del Curso: Parte 3A — Respuesta del Sistema: Linealidad de Amplitud, Series de Fourier, Ancho de Banda, Fase Lineal y Distorsión
Arquitectura / Controlador Estudiado: Superposición, Descomposición Espectral de Fourier, Función H(jω), Ancho de Banda a -3 dB, Retardo de Grupo y Distorsión THD

$$x(t) = a_0 + \Sigma[a_n \cdot \cos(n\omega_0 t) + b_n \cdot \sin(n\omega_0 t)] \mid |H(j\omega)|_{dB} = 20 \cdot \log 10|H(j\omega)| \mid \tau_g = -d\varphi(\omega)/d\omega \mid THD = \sqrt{(\Sigma V_n^2)} / V_1$$

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Relación entrada-salida y linealidad de amplitud: cumplimiento simultáneo de proporcionalidad (homogeneidad) y aditividad (superposición).
- Representación de señales periódicas mediante Series de Fourier: componente continua, frecuencia fundamental ω_0 y espectros discretos de amplitud y fase (onda cuadrada 1/n frente a triangular $1/n^2$).
- Respuesta en frecuencia H(jω), diagramas de Bode en decibelios, frecuencia de corte de media potencia (-3 dB o 0.707 de amplitud) y definición de ancho de banda en filtros paso bajo, paso alto y paso banda.
- Linealidad de fase φ(ω) = -ω·t_d para retardo de grupo constante τ_g, y distinción entre distorsión lineal (atenuación y fase) y distorsión no lineal por saturación (generación de armónicos nuevos, THD e intermodulación).

2. Parámetros Nominales de Configuración del Controlador (Video 05)

Parámetro del Controlador / Interfase	Símbolo	Valor Nominal	Función en el Lazo de Control / Automatización
Ganancia Proporcional	`Kp (PB = 100/Kp)`	3.2	Esfuerzo inmediato proporcional al error actual e(t)
Tiempo Integral	`Ti [s]`	0.4 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]`	0.5 ms	Intervalo determinista de conversión ADC, cálculo discreto y actualización PWM
Límites de Saturación Anti-Windup	`[u_min, u_max]`	[-10, 10]	Cota física del actuador para integración condicional (Clamping)
Rango Escalado de Ingeniería (SCALE_X)	`[PV_min, PV_max]`	[-5, 5]	Conversión lineal desde cuentas crudas del ADC (0..1023 o 0..27648)

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Introducción a la Respuesta del Sistema en Electrónica

Relación Entrada-Salida x(t) -> y(t), Dominio del Tiempo frente a Dominio de la Frecuencia y Fidelidad de Transmisión

- **Turno 1 (Masculina):** En ingeniería electrónica e instrumentación modelamos todo circuito, filtro, sensor o amplificador como un sistema que recibe una señal de excitación equis de te en su entrada y entrega una señal de respuesta ye de te en su salida.
- **Turno 2 (Femenina):** La pregunta central del análisis de respuesta de sistemas es determinar con exactitud cómo el circuito modifica la amplitud, la fase y la forma temporal de las señales que lo atraviesan.
- **Turno 3 (Masculina):** Este estudio se realiza de manera complementaria en dos dominios: en el dominio del tiempo evaluamos la respuesta transitoria ante cambios bruscos como un escalón, y en el dominio de la frecuencia analizamos cómo responde el sistema ante componentes senoidales.
- **Turno 4 (Femenina):** Para que una cadena electrónica transmita o amplifique una señal compleja sin deformarla, debe satisfacer simultáneamente criterios estrictos de linealidad de amplitud, ancho de banda suficiente y linealidad de fase.

Diapositiva 02: Linealidad de Amplitud: Proporcionalidad y Superposición

Principio de Homogeneidad f(a·x) = a·f(x), Aditividad f(x_1 + x_2) = f(x_1) + f(x_2) y Rango Dinámico Lineal

- **Turno 1 (Masculina):** Decimos que un sistema electrónico posee linealidad de amplitud cuando su relación entrada-salida cumple rigurosamente el principio de superposición, compuesto por las propiedades de homogeneidad y aditividad.
- **Turno 2 (Femenina):** La propiedad de homogeneidad o proporcionalidad establece que si multiplicamos la amplitud de la señal de entrada por una constante escalar alfa, la salida se multiplica exactamente por la misma constante alfa.
- **Turno 3 (Masculina):** La propiedad de aditividad garantiza que si aplicamos en la entrada la suma de dos señales simultáneas equis uno más equis dos, la respuesta en la salida es la suma exacta de las respuestas individuales sin que una señal interfiera con la otra.
- **Turno 4 (Femenina):** Una propiedad fundamental de todo sistema lineal invariante en el tiempo es que al ser excitado por una sinusoide pura de frecuencia omega, su salida en régimen permanente es otra sinusoide pura de idéntica frecuencia omega, cambiando únicamente su amplitud y su fase.

Diapositiva 03: Representación de Señales en Series de Fourier

Teorema de Fourier para Señales Periódicas de Periodo T_0, Frecuencia Fundamental ω_0 = 2π/T_0 y Coeficientes a_0, a_n y b_n

- **Turno 1 (Masculina):** La mayoría de las señales prácticas en electrónica, como pulsos de reloj, ondas cuadradas, señales de sensores o audio, no son simples ondas senoidales puras.
- **Turno 2 (Femenina):** El teorema de Fourier demuestra que cualquier señal periódica equis de te con periodo te sub cero puede descomponerse de manera exacta como la suma de un valor medio continuo más una serie infinita de senos y cosenos armónicamente relacionados.
- **Turno 3 (Masculina):** La primera componente oscilatoria tiene la frecuencia fundamental omega sub cero igual a dos pi sobre el periodo, mientras que los armónicos superiores ocurren en múltiplos enteros ene veces omega sub cero.
- **Turno 4 (Femenina):** Combinando cada par de coseno y seno del mismo orden ene obtenemos la forma trigonométrica compacta en amplitud ce sub ene y fase fi sub ene, la cual nos permite aplicar superposición armónico por armónico a través de cualquier circuito lineal.

Diapositiva 04: Espectros de Frecuencia de Ondas Típicas en Electrónica

Espectro Discreto de Líneas, Onda Cuadrada (Armónicos Impares 1/n), Onda Triangular (1/n^2) y Relación entre Flancos y Alta Frecuencia

- **Turno 1 (Masculina):** Al graficar la amplitud ce sub ene y la fase fi sub ene de cada armónico frente a la frecuencia obtenemos el espectro en frecuencia de la señal, el cual revela cómo se distribuye su energía espectral.
- **Turno 2 (Femenina):** Por ejemplo, una onda cuadrada simétrica carece de armónicos pares y contiene únicamente armónicos impares primero, tercero, quinto y séptimo, cuyas amplitudes decrecen lentamente en proporción a uno sobre ene.
- **Turno 3 (Masculina):** En cambio, una onda triangular continua tiene transiciones suaves sin saltos abruptos, por lo que sus armónicos impares decaen mucho más rápido, en proporción a uno sobre ene al cuadrado.
- **Turno 4 (Femenina):** Esta comparación nos enseña una regla física universal: cuanto más rápidos y verticales sean los flancos de subida y bajada de una señal en el tiempo, mayor será el contenido de armónicos de alta frecuencia que el circuito deberá transmitir.

Diapositiva 05: Respuesta en Frecuencia del Sistema y Diagramas de Bode

Función de Transferencia Sinusoidal H(jω) = |H(jω)|·e^(jφ(ω)), Ganancia en Decibelios 20·log10|H| y Escala Logarítmica

- **Turno 1 (Masculina):** La respuesta en frecuencia de un sistema lineal se describe mediante la función compleja hache de jota omega, definida como el cociente entre el fasor de salida y el fasor de entrada para cada frecuencia angular omega.
- **Turno 2 (Femenina):** Su módulo representa la ganancia de amplitud del sistema a esa frecuencia, mientras que su argumento fi de omega indica el desplazamiento de fase que sufre ese armónico al atravesar el circuito.
- **Turno 3 (Masculina):** Para visualizar el comportamiento a lo largo de varios órdenes de magnitud utilizamos el diagrama de Bode, el cual grafica la ganancia en decibelios igual a veinte por el logaritmo base diez del módulo frente a la frecuencia en escala logarítmica.
- **Turno 4 (Femenina):** En la escala de decibelios, una ganancia unitaria equivale a cero decibelios, reducir la amplitud de tensión a uno sobre raíz de dos equivale a menos tres decibelios, y dividir la amplitud por diez corresponde a menos veinte decibelios.

Diapositiva 06: Ancho de Banda y Clasificación de Filtros Electrónicos

Punto de Media Potencia (-3 dB o 0.707 del Máximo), Ancho de Banda BW = f_H - f_L y Filtros Paso Bajo, Paso Alto y Paso Banda

- **Turno 1 (Masculina):** Definimos el ancho de banda de un amplificador o filtro como el intervalo de frecuencias dentro del cual la ganancia de amplitud se mantiene por encima del setenta punto siete por ciento de su valor máximo en banda de paso.
- **Turno 2 (Femenina):** Dado que la potencia es proporcional al cuadrado del voltaje, una caída de amplitud a uno sobre raíz de dos reduce la potencia entregada exactamente a la mitad, razón por la cual las frecuencias límite de menos tres decibelios se llaman frecuencias de corte o de media potencia.
- **Turno 3 (Masculina):** En un sistema paso bajo que amplifica desde corriente continua cero hercios hasta una frecuencia de corte superior efe sub ce, el ancho de banda es directamente igual a efe sub ce.
- **Turno 4 (Femenina):** En un amplificador acoplado en alterna o en un filtro paso banda, el ancho de banda es la diferencia entre la frecuencia de corte superior efe sub hache y la frecuencia de corte inferior efe sub ele.

Diapositiva 07: Linealidad de Fase y Retardo de Grupo Constante

Desplazamiento Temporal Δt en una Sinusoide, Condición de Fase Lineal φ(ω) = -ω·t_d y Retardo de Grupo τ_g = -dφ/dω

- **Turno 1 (Masculina):** Para preservar intacta la forma de una onda compuesta por múltiples armónicos no basta con que todos reciban la misma ganancia de amplitud; también deben experimentar exactamente el mismo retardo de tiempo te sub de al viajar por el circuito.
- **Turno 2 (Femenina):** Si una componente de frecuencia omega se retrasa un tiempo fijo te sub de segundos, su expresión coseno de omega por te menos te sub de equivale a un desfase angular fi de omega igual a menos omega por te sub de.
- **Turno 3 (Masculina):** Esto demuestra que un retardo de tiempo constante para todas las frecuencias exige que el ángulo de fase fi de omega varíe de forma estrictamente lineal y proporcional con la frecuencia.
- **Turno 4 (Femenina):** Llamamos retardo de grupo tau sub ge a menos la derivada de la fase respecto a la frecuencia angular; cuando la fase es lineal con la frecuencia, el retardo de grupo es constante y todos los armónicos llegan sincronizados a la salida.

Diapositiva 08: Distorsión de Señales I: Distorsión de Amplitud y Distorsión de Fase

Efectos de un Ancho de Banda Insuficiente (Redondeo de Flancos y Sobrepicos) y Dispersión Temporal por Fase No Lineal

- **Turno 1 (Masculina):** Cuando un sistema lineal incumple las condiciones ideales de banda plana o fase lineal, la señal sufre distorsión lineal, la cual no crea frecuencias nuevas pero altera la proporción o la alineación de los armónicos existentes.
- **Turno 2 (Femenina):** La distorsión de amplitud o de atenuación ocurre cuando la ganancia del sistema no es constante dentro del espectro de la señal: si atenuamos los armónicos altos de una onda cuadrada, los flancos se vuelven lentos y redondeados.
- **Turno 3 (Masculina):** Si por el contrario atenuamos las bajas frecuencias mediante un acoplamiento capacitivo excesivo, el techo plano del pulso se inclina hacia cero produciendo caída de meseta o sag.
- **Turno 4 (Femenina):** Por su parte, la distorsión de fase aparece cuando el ángulo de fase no es proporcional a la frecuencia: los distintos armónicos viajan a velocidades diferentes y se desalinean al sumarse, deformando asimétricamente los pulsos.

Diapositiva 09: Distorsión de Señales II: Distorsión No Lineal, Armónica (THD) e Intermodulación

Curvatura de la Característica de Transferencia, Recorte por Saturación (Clipping), Generación de Armónicos Nuevos y THD%

- **Turno 1 (Masculina):** La distorsión no lineal es fundamentalmente distinta y más severa: ocurre cuando el circuito opera fuera de su rango lineal por curvatura de transistores o por recorte de saturación, violando el principio de superposición.
- **Turno 2 (Femenina):** Al aplicar una sinusoide pura de frecuencia efe sub cero a un amplificador no lineal, la salida deja de ser senoidal y aparecen componentes espectrales nuevas en múltiplos dos efe cero, tres efe cero y superiores que no existían en la entrada.
- **Turno 3 (Masculina):** Cuantificamos este fenómeno mediante la Distorsión Armónica Total o THD, definida como la raíz cuadrada de la suma de los cuadrados de las tensiones eficaces de los armónicos superiores dividida entre la tensión eficaz de la fundamental.
- **Turno 4 (Femenina):** Además, si ingresan dos frecuencias simultáneas efe uno y efe dos a un elemento no lineal, se generan productos de intermodulación en las frecuencias suma y diferencia, causando interferencia cruzada entre canales.

Diapositiva 10: Síntesis de la Parte 3A y Conexión con la Dinámica de Sistemas

Integración de Superposición, Series de Fourier, Ancho de Banda y Fase Lineal hacia los Modelos de Orden 0, 1 y 2

- **Turno 1 (Masculina):** En esta primera mitad de la Parte tres hemos establecido cómo el teorema de Fourier y la función de respuesta en frecuencia permiten predecir el comportamiento de cualquier circuito lineal ante señales arbitrarias.
- **Turno 2 (Femenina):** Comprendimos que la reproducción fiel de una forma de onda exige tres requisitos: linealidad de amplitud para evitar distorsión armónica, ganancia plana dentro del ancho de banda y fase lineal con la frecuencia.
- **Turno 3 (Masculina):** Ahora bien, ¿qué determina físicamente la forma de la respuesta en frecuencia y de la respuesta transitoria de un circuito o sensor?
- **Turno 4 (Femenina):** Lo determina el número de elementos independientes almacenadores de energía, lo que nos lleva en la Parte tres B al estudio exhaustivo de los sistemas de orden cero, primer orden y segundo orden.

4. Firmware C++/MCU: PID Discreto Tustin, FSM y CRC-16 (controlador_tiempo_real_video_05.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 05: Video 05 — Parte 3A: Respuesta del Sistema, Linealidad de Amplitud, Series de Fourier, Ancho de Banda, Linealidad de Fase y Distorsión
 * Archivo: controlador_tiempo_real_video_05.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:
 *   Superposición, Descomposición Espectral de Fourier, Función H(jω), Ancho de Banda a -3 dB, Retardo de Grupo y Distorsión THD
 * ============================================================================
 */

#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 05) ---
const float KP_GAIN      = 3.200f;
const float TI_SEC       = 0.400f;
const float TD_SEC       = 0.060f;
const float N_FILT       = 15.0f;
const float TS_SEC       = 0.0005f; // Periodo de muestreo Ts = 0.5 ms
const float U_MIN_LIM    = -10.0f;
const float U_MAX_LIM    = 10.0f;
const float PV_MIN_ENG   = -5.0f;
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 05"));
  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_05.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
VIDEO 05: Video 05 — Parte 3A: Respuesta del Sistema, Linealidad de Amplitud, Series de Fourier, Ancho de Banda,
Linealidad de Fase y Distorsión
Archivo: supervision_sbc_y_sintonizacion_video_05.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 05
KP_NOMINAL = 3.2
TI_NOMINAL = 0.4
TD_NOMINAL = 0.06
N_FILTRO   = 15
TS_SEG     = 0.0005
U_MIN      = -10
U_MAX      = 10


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 05")
    print(" Enfoque:", "Superposición, Descomposición Espectral de Fourier, Función H(jω), Ancho de Banda a -3 dB,
Retardo de Grupo y Distorsión THD")
    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_05.st)

```
(* ================================================================================
 * CURSO COMPLETO: FUNDAMENTOS DE ELECTRÓNICA
 * VIDEO 05: Video 05 — Parte 3A: Respuesta del Sistema, Linealidad de Amplitud, Series de Fourier, Ancho de
Banda, Linealidad de Fase y Distorsión
 * Archivo: automatizacion_plc_iec61131_video_05.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_V05) 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_V05
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 := 1.0;
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 := -5.0;
    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 := 3.200;
    rTi_s                : REAL := 0.400;
    rTd_s                : REAL := 0.060;
```

```
    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
```