CURSO COMPLETO: SISTEMAS DE COMUNICACIÓN Y REDES INALÁMBRICAS EN MECATRÓNICA

Video 04 — Parte 2B: ZigBee, LoRa / LoRaWAN y Otros Protocolos Inalámbricos en Mecatrónica — IEEE 802.15.4 DSSS, ZigBee 3.0 AODV, LoRa CSS (SF7-SF12), LoRaWAN (Clases A/B/C), Thread/Matter, WirelessHART, UWB y 5G URLLC

Parte del Curso: Parte 2B — ZigBee, LoRa / LoRaWAN y Otros Protocolos Inalámbricos en Mecatrónica
Sistemas / Protocolos Estudiados: IEEE 802.15.4 DSSS, Malla ZigBee 3.0 (AODV), Modulación LoRa CSS (SF7-SF12), LoRaWAN, Thread/Matter, WirelessHART, UWB y 5G URLLC

```
Rs_LoRa = BW / 2^SF | Rb_LoRa = SF · (BW / 2^SF) · (4 / (4 + CR)) | d_UWB = c · (T_round - T_reply) / 2
```

1. Síntesis de Conceptos Clave y Reglas de Ingeniería

- Estándar IEEE 802.15.4 (capa física DSSS de 32 chips por símbolo a 250 kbps y acceso CSMA/CA con tramas de 127 bytes) y arquitectura ZigBee 3.0 (Coordinador, Router y End Device) con enrutamiento dinámico autocurativo AODV en malla.
- Tecnología LoRa (Long Range) en bandas Sub-GHz mediante modulación Chirp Spread Spectrum (CSS con Up-Chirps y Down-Chirps) capaz de demodular señales hasta 20 dB por debajo del piso de ruido térmico; ajuste de Spreading Factor (SF7 a SF12 ortogonales), ancho de banda, Coding Rate y cálculo del Tiempo en el Aire (Time-on-Air).
- Arquitectura LoRaWAN en estrella de estrellas (End Nodes, Gateways, Network Server con ADR y Application Server), operación de dispositivos Clase A (ventanas RX1/RX2), Clase B (Beacons) y Clase C (escucha continua) y doble cifrado AES-128 (NwkSKey + AppSKey).
- Protocolos IPv6 en malla (6LoWPAN, Thread con Border Routers y capa de aplicación Matter), estándares industriales deterministas de proceso (WirelessHART e ISA100.11a con malla TDMA en ranuras de 10 ms y salto de canal TSCH con confiabilidad >99.9%), trazabilidad RFID/NFC, posicionamiento interior centimétrico Ultra-Wideband (UWB IEEE 802.15.4z por Tiempo de Vuelo ToF) y redes celulares LTE-M, NB-IoT y 5G Industrial URLLC (latencia <= 1 ms y fiabilidad 99.9999%).

2. Parámetros Nominales de Diseño de Red y Enlace RF (Video 04)

Parámetro de Comunicación / Radiofrecuencia	Símbolo	Valor Nominal	Función en la Red Industrial / Enlace Inalámbrico
Velocidad de Transmisión en Serie	`Baud Rate / Rb`	250000 bps	Caudal de transferencia en bus físico UART / RS-485 / CAN
Ancho de Banda de Canal	`B [kHz]`	125 kHz	Determina la capacidad de Shannon-Hartley y tasa de símbolos
Relación Señal a Ruido	`SNR [dB]`	-12 dB	Margen sobre el piso de ruido térmico e interferencia industrial
Frecuencia Portadora ISM y Potencia TX	`f [MHz] / Pt [dBm]`	915 MHz / +17 dBm	Cálculo de longitud de onda λ, antena λ/4 y ecuación de Friis
Factor de Ensanchamiento LoRa y Alcance	`SF / d [m]`	SF10 / 4500 m	Compromiso entre sensibilidad de recepción y Tiempo en el Aire
Watchdog de Comunicación Determinista	`T_timeout [ms]`	100 ms	Ventana máxima antes de conmutar actuadores a estado Fail-Safe

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Estándar IEEE 802.15.4: Capa Física DSSS y Acceso CSMA/CA

Los 16 Canales en 2.4 GHz (250 kbps), Ensanchamiento por Secuencia Directa (32 Chips) y Tramas de 127 Bytes

- **Turno 1 (Masculina):** Una gran familia de protocolos inalámbricos de instrumentación y automatización — incluyendo ZigBee, Thread, seis LoWPAN, WirelessHART e ISA cien punto once a— comparte un mismo cimiento de ingeniería: el estándar IEEE ochocientos dos punto quince punto cuatro.
- **Turno 2 (Femenina):** Esta norma define únicamente la Capa Física y la Capa de Enlace de Datos para redes de bajo consumo, dividiendo la banda de dos punto cuatro gigahercios en dieciséis canales numerados del once al veintiséis con una velocidad bruta de doscientos cincuenta kilobits por segundo.
- **Turno 3 (Masculina):** Para resistir el ruido industrial, su capa física utiliza Espectro Ensanchado por Secuencia Directa, o DSSS: cada símbolo de cuatro bits se transforma en una secuencia pseudoaleatoria de treinta y dos chips modulados en fase en cuadratura con desplazamiento.
- **Turno 4 (Femenina):** En la subcapa MAC, los nodos emplean acceso múltiple con prevención de colisiones ce ese eme a barra ce a evaluando primero la energía libre del canal, y transmiten tramas compactas de hasta ciento veintisiete bytes con acuse de recibo inmediato en hardware.

Diapositiva 02: Arquitectura ZigBee 3.0: Coordinador, Router, End Device y Malla AODV

Capas NWK/APS sobre IEEE 802.15.4, Biblioteca de Clústeres ZCL y Ruteo Dinámico Autocurativo (Self-Healing)

- **Turno 1 (Masculina):** Sobre los cimientos físicos de IEEE ochocientos dos punto quince punto cuatro, la especificación ZigBee tres punto cero construye las capas de Red y de Aplicación para formar redes en malla autoorganizadas de hasta sesenta y cinco mil nodos.
- **Turno 2 (Femenina):** Toda red ZigBee se compone de tres roles jerárquicos: un único Coordinador que inicializa la red y administra las claves criptográficas, múltiples Routers alimentados continuamente que reenvían paquetes, y los Dispositivos Finales o End Devices de bajo consumo.
- **Turno 3 (Masculina):** La fortaleza industrial de ZigBee reside en su algoritmo de enrutamiento vector de distancia bajo demanda, o AODV: cuando un nodo necesita enviar un dato, descubre la ruta de menor costo a través de los Routers intermedios.
- **Turno 4 (Femenina):** Si una máquina metálica se interpone o un Router pierde energía, la malla ZigBee detecta la falta de confirmación y recalcula automáticamente un camino alternativo en milisegundos, propiedad conocida como red autocurativa o Self-Healing Mesh.

Diapositiva 03: Tecnología LoRa (Long Range): Modulación Chirp Spread Spectrum (CSS)

Barridos Lineales de Frecuencia (Up-Chirp / Down-Chirp) y Demodulación hasta 20 dB por Debajo del Piso de Ruido

- **Turno 1 (Masculina):** ¿Cómo podemos enlazar estaciones de bombeo, sensores agrícolas o vehículos autónomos situados a cinco, diez o quince kilómetros de distancia usando apenas unos miliamperios de batería? Mediante la tecnología de largo alcance LoRa.
- **Turno 2 (Femenina):** LoRa define una capa física patentada que opera en las bandas sub-gigahercio libres —cuatrocientos treinta y tres, ochocientos sesenta y ocho o novecientos quince megahercios— utilizando modulación de espectro ensanchado por pulsos de barrido en frecuencia, llamada Chirp Spread Spectrum.
- **Turno 3 (Masculina):** Un Chirp es una señal sinusoidal cuya frecuencia aumenta continuamente en el tiempo, llamada Up-Chirp, o disminuye linealmente, llamada Down-Chirp, barriendo todo el ancho de banda del canal para codificar cada símbolo de información.
- **Turno 4 (Femenina):** Cuando el receptor aplica correlación matemática multiplicando la señal entrante por un Chirp conjugado, toda la energía dispersa se concentra en un pico nítido en la transformada rápida de Fourier, permitiendo decodificar señales veinte decibelios por debajo del ruido térmico.

Diapositiva 04: Parámetros Físicos de LoRa: Spreading Factor (SF7 a SF12), BW y CR

Relación Rs = BW / 2^SF, Ortogonalidad entre Factores de Ensanchamiento, Tiempo en el Aire y Duty Cycle

- **Turno 1 (Masculina):** Al programar un transceptor LoRa como el ese equis doce sesenta y dos o doce setenta y seis, el ingeniero ajusta tres parámetros físicos fundamentales: el Factor de Ensanchamiento o Spreading Factor, el Ancho de Banda y la Tasa de Codificación.
- **Turno 2 (Femenina):** El Factor de Ensanchamiento, abreviado ese efe, toma valores enteros desde ese efe siete hasta ese efe doce, de modo que cada símbolo está formado por dos elevado a la ese efe chips: en ese efe siete son ciento veintiocho chips y en ese efe doce son cuatro mil noventa y seis.
- **Turno 3 (Masculina):** En un canal estándar de ciento veinticinco kilohercios, usar ese efe siete brinda la máxima velocidad de unos cinco punto cinco kilobits por segundo y un tiempo en el aire de apenas cincuenta milisegundos, ideal para nodos cercanos.
- **Turno 4 (Femenina):** En cambio, subir a ese efe doce incrementa el tiempo en el aire a más de un segundo y reduce la tasa a trescientos bits por segundo, pero gana doce decibelios adicionales de sensibilidad alcanzando menos ciento treinta y siete de be eme con total ortogonalidad entre factores.

Diapositiva 05: Arquitectura de Red LoRaWAN: Clases A, B y C, Gateways y AES-128

Topología Estrella de Estrellas, Adaptive Data Rate (ADR), Activación OTAA y Ventanas de Recepción RX1/RX2

- **Turno 1 (Masculina):** Mientras que LoRa es la capa física de modulación por radio, LoRaWAN es el protocolo estandarizado de capa de enlace MAC y arquitectura de red que organiza miles de sensores en una topología de estrella de estrellas.
- **Turno 2 (Femenina):** Cuando un nodo final transmite una trama LoRaWAN, esta es escuchada simultáneamente por todas las pasarelas o Gateways cercanas, las cuales reenvían el paquete por IP hacia el Servidor de Red que filtra duplicados y ajusta la potencia y el factor ese efe mediante ADR.
- **Turno 3 (Masculina):** LoRaWAN define tres clases de dispositivos según su estrategia de escucha descendente: los nodos Clase A son los de menor consumo, pues solo abren dos breves ventanas de recepción erre equis uno y erre equis dos inmediatamente después de haber enviado un paquete ascendente.
- **Turno 4 (Femenina):** Los nodos Clase B abren ventanas adicionales sincronizadas mediante balizas de tiempo del Gateway, mientras que los nodos Clase C mantienen el receptor encendido continuamente para accionar válvulas o relés sin retardo, todo bajo doble cifrado AES de ciento veintiocho bits.

Diapositiva 06: Otros Protocolos en Malla Basados en IPv6: 6LoWPAN, Thread y Matter

Compresión de Cabeceras IPv6 sobre IEEE 802.15.4, Enrutadores de Borde sin SPOF e Interoperabilidad de Aplicación

- **Turno 1 (Masculina):** Además de los cuatro grandes estándares inalámbricos, el ingeniero mecatrónico debe conocer otros protocolos especializados del estado del arte, comenzando por aquellos que llevan el direccionamiento IP versión seis nativo hasta los sensores en malla.
- **Turno 2 (Femenina):** El estándar seis LoWPAN introduce una capa inteligente de adaptación y compresión de encabezados que reduce los cuarenta bytes de una cabecera IPv6 a tan solo dos o tres bytes, permitiendo que viajen dentro de las pequeñas tramas de IEEE ochocientos dos punto quince punto cuatro.
- **Turno 3 (Masculina):** Sobre seis LoWPAN se construye el protocolo Thread, una red en malla segura y autocurativa que elimina el punto único de falla del coordinador clásico permitiendo múltiples Enrutadores de Borde redundantes hacia la red Ethernet o Wi-Fi.
- **Turno 4 (Femenina):** En la capa de aplicación superior converge el estándar Matter, que unifica el modelo de datos de sensores y actuadores para que operen de manera idéntica y local tanto sobre redes Thread de bajo consumo como sobre enlaces Wi-Fi de alta velocidad.

Diapositiva 07: Protocolos Inalámbricos Industriales Deterministas: WirelessHART e ISA100.11a

Superación de CSMA/CA mediante Malla Sincronizada TDMA (Ranuras de 10 ms), TSCH y Listas Negras de Canales

- **Turno 1 (Masculina):** En refinerías, plantas químicas, acerías y calderas industriales, la pérdida o el retraso aleatorio de una trama inalámbrica es inaceptable; allí dominan dos estándares industriales deterministas: WirelessHART, norma IEC sesenta y dos mil quinientos noventa y uno, e ISA cien punto once a.
- **Turno 2 (Femenina):** A diferencia de ZigBee que usa contienda probabilística, WirelessHART e ISA cien punto once a implementan una malla sincronizada por Acceso Múltiple por División de Tiempo, o te de eme a, dividiendo el tiempo en supertramas con ranuras exactas de diez milisegundos.
- **Turno 3 (Masculina):** En cada ranura de diez milisegundos, los nodos ejecutan además Salto de Canal Sincronizado en el Tiempo, o TSCH: cada transacción ocurre en una frecuencia distinta pseudoaleatoria y el gestor de red bloquea en lista negra los canales con interferencia persistente.
- **Turno 4 (Femenina):** Combinando esta diversidad en tiempo y en frecuencia con rutas espaciales redundantes en malla donde cada transmisor de presión o caudal dispone de al menos dos caminos físicos, estos protocolos alcanzan una confiabilidad industrial comprobada del noventa y nueve punto nueve por ciento.

Diapositiva 08: Protocolos de Identificación y Localización: RFID, NFC y Ultra-Wideband (UWB)

Trazabilidad Pasiva (LF/HF/UHF EPC Gen2) y Posicionamiento Centimétrico por Tiempo de Vuelo (ToF / TDoA) en UWB

- **Turno 1 (Masculina):** Para la trazabilidad automática de piezas en celdas de manufactura flexible y la navegación precisa de vehículos guiados en interiores donde no llega la señal satelital GPS, empleamos tecnologías de identificación por radiofrecuencia y banda ultraancha.
- **Turno 2 (Femenina):** Los sistemas RFID y ene efe ce utilizan etiquetas pasivas sin batería que se energizan con el campo del lector: en trece punto cincuenta y seis megahercios por acoplamiento inductivo cercano y en ultra-alta frecuencia UHF hasta diez metros por retrodispersión.
- **Turno 3 (Masculina):** Por otro lado, cuando intentamos calcular la distancia a un robot midiendo la intensidad RSSI de Wi-Fi o Bluetooth, los rebotes metálicos producen errores de varios metros; este problema lo resuelve de raíz el protocolo Ultra-Wideband o UWB, norma IEEE ochocientos dos punto quince punto cuatro zeta.
- **Turno 4 (Femenina):** En lugar de medir potencia, UWB transmite pulsos ultracortos de menos de dos nanosegundos sobre un ancho de banda de quinientos megahercios y cronometra el Tiempo de Vuelo de ida y vuelta de la onda, logrando localización en tiempo real con precisión de cinco centímetros.

Diapositiva 09: Comunicaciones Celulares Mecatrónicas: LTE-M, NB-IoT y 5G Industrial URLLC

Espectro Licenciado de Área Amplia, Movilidad vs. Penetración Profunda y Lazos de Control a <= 1 ms con 99.9999% de Fiabilidad

- **Turno 1 (Masculina):** Completamos el panorama de protocolos inalámbricos con las tecnologías celulares sobre espectro licenciado, las cuales eliminan las interferencias de las bandas libres y se dividen en soluciones masivas de bajo consumo y redes privadas de ultra-baja latencia.
- **Turno 2 (Femenina):** Para telemetría de área amplia disponemos de ele te e eme, que ofrece hasta un megabit por segundo con soporte de movilidad para maquinaria pesada y flotas, y ene be i o te, que utiliza apenas doscientos kilohercios de ancho de banda con máxima penetración subterránea.
- **Turno 3 (Masculina):** En el interior de las fábricas inteligentes irrumpe el cinco G Industrial Privado con sus tres perfiles: banda ancha móvil mejorada para visión artificial, comunicaciones masivas de máquinas y, de forma estelar para la mecatrónica, el perfil URLLC.
- **Turno 4 (Femenina):** Las comunicaciones ultra-confiables de baja latencia URLLC, integradas con redes sensibles al tiempo te ese ene, garantizan latencias de aire menores o iguales a un milisegundo y una fiabilidad de seis nueves, permitiendo cerrar lazos de control y paros de emergencia inalámbricos.

Diapositiva 10: Matriz Comparativa de Selección Inalámbrica y Transición al Cierre

Metodología de Decisión en 5 Dimensiones (Alcance, Caudal, Energía, Latencia y Topología) hacia el Resumen Final

- **Turno 1 (Masculina):** Al concluir la Parte dos B disponemos de una metodología rigurosa en cinco dimensiones para seleccionar el protocolo inalámbrico óptimo en cualquier diseño mecatrónico: alcance requerido, tasa de datos, presupuesto energético, determinismo temporal y topología.
- **Turno 2 (Femenina):** Para sensores portátiles y periféricos de máquina elegimos Bluetooth Low Energy; para mallas densas de automatización elegimos ZigBee tres punto cero o Thread con Matter; y para instrumentación crítica de proceso en planta elegimos WirelessHART o ISA cien punto once a.
- **Turno 3 (Masculina):** Cuando la aplicación exige cubrir kilómetros con mínimo consumo optamos por LoRaWAN o ene be i o te; para localización centimétrica de robots usamos banda ultraancha UWB; y para alto caudal o lazos de control de un milisegundo desplegamos Wi-Fi seis o cinco G URLLC.
- **Turno 4 (Femenina):** Con todos los bloques del curso completados, acompáñennos en el siguiente módulo para realizar la síntesis ejecutiva e integradora de los sistemas de comunicación y redes inalámbricas en mecatrónica.

4. Firmware C++/MCU: Trama Industrial, CRC-16 y FSM Fail-Safe (comunicacion_embebida_video_04.ino)

```
/*
 * ============================================================================
 * CURSO COMPLETO: SISTEMAS DE COMUNICACIÓN Y REDES INALÁMBRICAS EN MECATRÓNICA
 * VIDEO 04: Video 04 — Parte 2B: ZigBee, LoRa / LoRaWAN y Otros Protocolos Inalámbricos en Mecatrónica
 * Archivo: comunicacion_embebida_video_04.ino
 * ----------------------------------------------------------------------------
 * Descripción Técnica:
 *   Firmware determinista en C/C++ Embebido para Microcontrolador (ESP32-S3 /
 *   STM32 / Arduino Industrial) que implementa:
 *     1. Empaquetado y análisis sintáctico de tramas binarias industriales
 *        compatibles con RS-485 (Modbus RTU) y pasarelas inalámbricas (BLE/LoRa).
 *     2. Cálculo polinómico de Verificación de Redundancia Cíclica CRC-16
 *        (polinomio reflectado 0xA001) para inmunidad ante ruido electromagnético.
 *     3. Control de dirección Half-Duplex (pin DE/RE de transceptor MAX485) y
 *        lectura de telemetría de radiofrecuencia (RSSI / SNR).
 *     4. Máquina de Estados Finitos (FSM) no bloqueante con supervisión de
 *        Timeout (100 ms) y estado seguro (Fail-Safe) ante pérdida de enlace.
 *
 * Enfoque del Módulo:
 *   IEEE 802.15.4 DSSS, Malla ZigBee 3.0 (AODV), Modulación LoRa CSS (SF7-SF12), LoRaWAN, Thread/Matter,
WirelessHART, UWB y 5G URLLC
 * ============================================================================
 */

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

// --- Definición de Pines de Interfase de Comunicación y Estado Seguro ---
const uint8_t PIN_RS485_DE_RE   = 4;  // Control de dirección Half-Duplex (HIGH = TX, LOW = RX)
const uint8_t PIN_CS_SPI_RADIO  = 10; // Chip Select (CS#) para transceptor SPI (LoRa SX1262 / CAN)
const uint8_t PIN_LED_ENLACE_OK = 2;  // Indicador de trama válida recibida
const uint8_t PIN_DO_FAILSAFE   = 8;  // Salida de actuador con corte seguro por Timeout

// --- Parámetros Nominales de Comunicación (Video 04) ---
const uint32_t BAUD_RATE_SERIE   = 250000UL;
const uint8_t  NODO_DIRECCION_ID = 0x01;       // Dirección de esclavo / nodo mecatrónico
const uint32_t TIMEOUT_ENLACE_MS = 100UL;      // Tiempo máximo sin trama antes de Fail-Safe
const float    FREQ_PORTADORA_MHZ = 915.0f;
const int8_t   TX_POWER_DBM       = 17;
const uint8_t  LORA_SPREADING_SF  = 10;

// --- Estructura de Trama Industrial Unificada (12 Bytes con CRC-16) ---
// [0]: SOF (0xAA) | [1]: Nodo ID | [2]: Función | [3..4]: Setpoint (uint16)
// [5..6]: Variable Proceso PV (uint16) | [7]: RSSI_dBm (int8) | [8]: EstadoFlags
// [9]: Secuencia | [10..11]: CRC-16-Modbus (Little-Endian)
struct TramaMecatronica {
  uint8_t  sof;
  uint8_t  nodoId;
  uint8_t  funcion;
  uint16_t consignaSp;
  uint16_t variablePv;
  int8_t   rssiDbm;
  uint8_t  estadoFlags;
  uint8_t  secuencia;
  uint16_t crc16;
};

enum EstadoEnlaceFSM : uint8_t {
  ENLACE_ESPERANDO_SOF = 0,
  ENLACE_RECIBIENDO_BYTES,
  ENLACE_VALIDANDO_CRC,
  ENLACE_EJECUTANDO_RESPUESTA,
  ENLACE_ERROR_FAILSAFE
};

EstadoEnlaceFSM g_estadoEnlace = ENLACE_ESPERANDO_SOF;
uint8_t         g_bufferRx[12];
```

```
uint8_t         g_indiceRx = 0;
uint32_t        g_ultimoPaqueteValidoMs = 0;
uint8_t         g_contadorSecuenciaTx = 0;
uint16_t        g_consignaActiva = 0;

// --- Cálculo Polinómico CRC-16-Modbus (Polinomio 0xA001) ---
uint16_t calcularCrc16Modbus(const uint8_t *datos, uint8_t longitud) {
  uint16_t crc = 0xFFFFU;
  for (uint8_t i = 0; i < longitud; i++) {
    crc ^= static_cast<uint16_t>(datos[i]);
    for (uint8_t bit = 0; bit < 8; bit++) {
      if ((crc & 0x0001U) != 0U) {
        crc = (crc >> 1U) ^ 0xA001U;
      } else {
        crc >>= 1U;
      }
    }
  }
  return crc;
}

// --- Transmisión Half-Duplex sobre Par Diferencial RS-485 / Enlace Serie ---
void transmitirTramaRespuesta(uint8_t funcion, uint16_t pvActual, int8_t rssiEstimado) {
  uint8_t tramaTx[12];
  tramaTx[0] = 0xAAU;
  tramaTx[1] = NODO_DIRECCION_ID;
  tramaTx[2] = funcion;
  tramaTx[3] = static_cast<uint8_t>(g_consignaActiva & 0xFFU);
  tramaTx[4] = static_cast<uint8_t>((g_consignaActiva >> 8U) & 0xFFU);
  tramaTx[5] = static_cast<uint8_t>(pvActual & 0xFFU);
  tramaTx[6] = static_cast<uint8_t>((pvActual >> 8U) & 0xFFU);
  tramaTx[7] = static_cast<uint8_t>(rssiEstimado);
  tramaTx[8] = (g_estadoEnlace == ENLACE_ERROR_FAILSAFE) ? 0x80U : 0x01U;
  tramaTx[9] = g_contadorSecuenciaTx++;

  uint16_t crc = calcularCrc16Modbus(tramaTx, 10);
  tramaTx[10] = static_cast<uint8_t>(crc & 0xFFU);
  tramaTx[11] = static_cast<uint8_t>((crc >> 8U) & 0xFFU);

  // Conmutar transceptor diferencial a modo Transmisión (DE/RE = HIGH)
  digitalWrite(PIN_RS485_DE_RE, HIGH);
  Serial.write(tramaTx, 12);
  Serial.flush(); // Esperar vaciado completo del registro de desplazamiento UART
  digitalWrite(PIN_RS485_DE_RE, LOW);
}

void setup() {
  pinMode(PIN_RS485_DE_RE, OUTPUT);
  pinMode(PIN_CS_SPI_RADIO, OUTPUT);
  pinMode(PIN_LED_ENLACE_OK, OUTPUT);
  pinMode(PIN_DO_FAILSAFE, OUTPUT);

  digitalWrite(PIN_RS485_DE_RE, LOW);   // Modo escucha por defecto en bus Half-Duplex
  digitalWrite(PIN_CS_SPI_RADIO, HIGH); // Línea CS# inactiva en nivel alto
  digitalWrite(PIN_DO_FAILSAFE, LOW);

  Serial.begin(BAUD_RATE_SERIE);
  g_ultimoPaqueteValidoMs = millis();
}

void loop() {
  uint32_t ahoraMs = millis();

  // 1. Procesamiento No Bloqueante de Bytes Entrantes en la FSM de Protocolo
  while (Serial.available() > 0) {
    uint8_t byteLeido = static_cast<uint8_t>(Serial.read());

    if (g_estadoEnlace == ENLACE_ESPERANDO_SOF || g_estadoEnlace == ENLACE_ERROR_FAILSAFE) {
```

```
      if (byteLeido == 0xAAU) {
        g_bufferRx[0] = byteLeido;
        g_indiceRx = 1;
        g_estadoEnlace = ENLACE_RECIBIENDO_BYTES;
      }
    } else if (g_estadoEnlace == ENLACE_RECIBIENDO_BYTES) {
      g_bufferRx[g_indiceRx++] = byteLeido;
      if (g_indiceRx >= 12U) {
        g_estadoEnlace = ENLACE_VALIDANDO_CRC;
      }
    }

    if (g_estadoEnlace == ENLACE_VALIDANDO_CRC) {
      uint16_t crcRecibido = static_cast<uint16_t>(g_bufferRx[10]) |
                             (static_cast<uint16_t>(g_bufferRx[11]) << 8U);
      uint16_t crcCalculado = calcularCrc16Modbus(g_bufferRx, 10);

      if (crcRecibido == crcCalculado && g_bufferRx[1] == NODO_DIRECCION_ID) {
        g_consignaActiva = static_cast<uint16_t>(g_bufferRx[3]) |
                           (static_cast<uint16_t>(g_bufferRx[4]) << 8U);
        g_ultimoPaqueteValidoMs = ahoraMs;
        digitalWrite(PIN_LED_ENLACE_OK, HIGH);
        digitalWrite(PIN_DO_FAILSAFE, HIGH);

        uint16_t pvSensorLocal = static_cast<uint16_t>(analogRead(A0));
        int8_t rssiMedido = -68; // Ejemplo de nivel RSSI en dBm (-68 dBm)
        transmitirTramaRespuesta(g_bufferRx[2], pvSensorLocal, rssiMedido);
      }
      g_indiceRx = 0;
      g_estadoEnlace = ENLACE_ESPERANDO_SOF;
    }
  }

  // 2. Supervisión Determinista de Watchdog de Comunicación (Timeout Fail-Safe)
  if ((ahoraMs - g_ultimoPaqueteValidoMs) > TIMEOUT_ENLACE_MS) {
    g_estadoEnlace = ENLACE_ERROR_FAILSAFE;
    g_consignaActiva = 0U;
    digitalWrite(PIN_LED_ENLACE_OK, LOW);
    digitalWrite(PIN_DO_FAILSAFE, LOW); // Desenergiza actuador por pérdida de red
  }
}
```

5. Script Python SBC: Shannon-Hartley, Presupuesto de Enlace Friis y LoRa CSS (analisis_redes_y_enlace_rf_video_04.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: SISTEMAS DE COMUNICACIÓN Y REDES INALÁMBRICAS EN MECATRÓNICA
VIDEO 04: Video 04 — Parte 2B: ZigBee, LoRa / LoRaWAN y Otros Protocolos Inalámbricos en Mecatrónica
Archivo: analisis_redes_y_enlace_rf_video_04.py
--------------------------------------------------------------------------------
Descripción Técnica:
  Herramienta de ingeniería en Python para Computadora de Placa Única (SBC /
  Raspberry Pi / Gateway IIoT) que ejecuta:
    1. Cálculo de Capacidad de Canal de Shannon-Hartley y Límite de Nyquist.
    2. Cálculo del Presupuesto de Enlace de Radiofrecuencia (Link Budget),
       Pérdida en el Espacio Libre (Ecuación de Friis / FSPL) y Margen de
       Desvanecimiento (Fade Margin) en bandas ISM (915 MHz).
    3. Cálculo de Tiempo en el Aire (Time-on-Air) y tasa de bits de la
       modulación Chirp Spread Spectrum (CSS) de LoRa (SF10).
    4. Empaquetado y verificación de tramas Modbus RTU con CRC-16 y payload
       JSON para telemetría MQTT / OPC UA.

Enfoque del Módulo:
  IEEE 802.15.4 DSSS, Malla ZigBee 3.0 (AODV), Modulación LoRa CSS (SF7-SF12), LoRaWAN, Thread/Matter,
WirelessHART, UWB y 5G URLLC
================================================================================
"""

import math
import struct
import json
from typing import Dict, Any

# --- Parámetros Nominales del Enlace Mecatrónico (Video 04) ---
BW_CANAL_KHZ      = 125.0
SNR_DB_NOMINAL    = -12.0
FREQ_ISM_MHZ      = 915.0
TX_POWER_DBM      = 17.0
DISTANCIA_METROS  = 4500.0
LORA_SF_NOMINAL   = 10


def calcular_capacidad_shannon_nyquist(bw_khz: float, snr_db: float, niveles_m: int = 4) -> Dict[str, float]:
    """Calcula la capacidad máxima de Shannon-Hartley y la tasa de Nyquist."""
    bw_hz = bw_khz * 1000.0
    snr_lineal = 10.0 ** (snr_db / 10.0)
    capacidad_shannon_bps = bw_hz * math.log2(1.0 + snr_lineal)
    capacidad_nyquist_bps = 2.0 * bw_hz * math.log2(niveles_m)
    return {
        "Ancho_Banda_kHz": round(bw_khz, 2),
        "SNR_dB": round(snr_db, 2),
        "SNR_Lineal": round(snr_lineal, 4),
        "Capacidad_Nyquist_kbps": round(capacidad_nyquist_bps / 1000.0, 3),
        "Capacidad_Shannon_kbps": round(capacidad_shannon_bps / 1000.0, 3),
    }


def calcular_presupuesto_enlace_friis(
    pt_dbm: float,
    freq_mhz: float,
    distancia_m: float,
    gt_dbi: float = 2.15,
    gr_dbi: float = 2.15,
    perdidas_obstaculos_db: float = 12.0,
    sensibilidad_rx_dbm: float = -118.0,
) -> Dict[str, Any]:
    """Evalúa la ecuación de Friis, FSPL y el Margen de Desvanecimiento (Fade Margin)."""
    distancia_km = max(distancia_m / 1000.0, 0.001)
    longitud_onda_m = 299.792458 / freq_mhz
```

```
    antena_cuarto_onda_cm = (longitud_onda_m / 4.0) * 100.0

    # FSPL(dB) = 20*log10(d_km) + 20*log10(f_MHz) + 32.44
    fspl_db = 20.0 * math.log10(distancia_km) + 20.0 * math.log10(freq_mhz) + 32.44
    pr_dbm = pt_dbm + gt_dbi + gr_dbi - fspl_db - perdidas_obstaculos_db
    fade_margin_db = pr_dbm - sensibilidad_rx_dbm

    return {
        "Frecuencia_MHz": freq_mhz,
        "Longitud_Antena_Cuarto_Onda_cm": round(antena_cuarto_onda_cm, 2),
        "FSPL_Espacio_Libre_dB": round(fspl_db, 2),
        "Potencia_Recibida_RSSI_dBm": round(pr_dbm, 2),
        "Margen_Desvanecimiento_dB": round(fade_margin_db, 2),
        "Enlace_Industrial_Viable": fade_margin_db >= 15.0,
    }


def calcular_parametros_fisicos_lora(sf: int, bw_khz: float = 125.0, cr_denom: int = 5, payload_bytes: int = 16) -
> Dict[str, float]:
    """Calcula velocidad de símbolo, velocidad de bits y Time-on-Air para LoRa CSS."""
    bw_hz = bw_khz * 1000.0
    chips_por_simbolo = 2 ** sf
    tasa_simbolos_rs = bw_hz / chips_por_simbolo
    t_simbolo_ms = (1.0 / tasa_simbolos_rs) * 1000.0
    tasa_bits_bps = sf * tasa_simbolos_rs * (4.0 / cr_denom)

    # Estimación estándar de símbolos de trama y Time-on-Air
    de_low_rate = 1 if (sf >= 11 and bw_khz <= 125.0) else 0
    num = 8 * payload_bytes - 4 * sf + 28 + 16
    den = 4 * (sf - 2 * de_low_rate)
    n_payload = 8 + max(math.ceil(num / den) * cr_denom, 0)
    tiempo_aire_ms = (12.25 + n_payload) * t_simbolo_ms

    return {
        "Spreading_Factor": sf,
        "Chips_por_Simbolo": chips_por_simbolo,
        "Tasa_Bits_Efectiva_bps": round(tasa_bits_bps, 2),
        "Tiempo_en_el_Aire_ms": round(tiempo_aire_ms, 2),
    }


def calcular_crc16_modbus(data: bytes) -> int:
    """Calcula el CRC-16-Modbus (polinomio reflectado 0xA001)."""
    crc = 0xFFFF
    for b in data:
        crc ^= b
        for _ in range(8):
            if crc & 0x0001:
                crc = (crc >> 1) ^ 0xA001
            else:
                crc >>= 1
    return crc & 0xFFFF


def construir_trama_modbus_rtu(slave_id: int, funcion: int, reg_inicio: int, valor_reg: int) -> bytes:
    """Construye una trama Modbus RTU completa con verificación CRC-16."""
    cabecera = struct.pack(">BBHH", slave_id, funcion, reg_inicio, valor_reg)
    crc = calcular_crc16_modbus(cabecera)
    return cabecera + struct.pack("<H", crc)


if __name__ == "__main__":
    print("=" * 78)
    print(" CURSO: SISTEMAS DE COMUNICACIÓN Y REDES INALÁMBRICAS — VIDEO 04")
    print(" Enfoque:", "IEEE 802.15.4 DSSS, Malla ZigBee 3.0 (AODV), Modulación LoRa CSS (SF7-SF12), LoRaWAN,
Thread/Matter, WirelessHART, UWB y 5G URLLC")
    print("=" * 78)
```

```
    cap = calcular_capacidad_shannon_nyquist(BW_CANAL_KHZ, SNR_DB_NOMINAL)
    print("\n[1] Capacidad de Canal (Nyquist y Shannon-Hartley):", json.dumps(cap, ensure_ascii=False))

    rf = calcular_presupuesto_enlace_friis(TX_POWER_DBM, FREQ_ISM_MHZ, DISTANCIA_METROS)
    print("\n[2] Presupuesto de Enlace RF (Ecuación de Friis):", json.dumps(rf, ensure_ascii=False))

    lora_cfg = calcular_parametros_fisicos_lora(LORA_SF_NOMINAL, BW_CANAL_KHZ)
    print("\n[3] Parámetros Físicos LoRa CSS:", json.dumps(lora_cfg, ensure_ascii=False))

    trama_rtu = construir_trama_modbus_rtu(slave_id=1, funcion=0x06, reg_inicio=0x0100, valor_reg=1500)
    print("\n[4] Trama Modbus RTU con CRC-16:", trama_rtu.hex(" ").upper())
```

6. Programa PLC IEC 61131-3 (ST + IL): Pasarela de Red, Heartbeat y Watchdog (pasarela_industrial_plc_video_04.st)

```
(* ============================================================================
 * CURSO COMPLETO: SISTEMAS DE COMUNICACIÓN Y REDES INALÁMBRICAS EN MECATRÓNICA
 * VIDEO 04: Video 04 — Parte 2B: ZigBee, LoRa / LoRaWAN y Otros Protocolos Inalámbricos en Mecatrónica
 * Archivo: pasarela_industrial_plc_video_04.st
 * ----------------------------------------------------------------------------
 * Estándar: IEC 61131-3 — Texto Estructurado (ST) + Equivalencia en Lista de
 *           Instrucciones (IL) para Pasarela de Comunicaciones Industriales
 * Descripción Técnica:
 *   Bloque de Función Industrial (FB_GestorRedMecatronica_V04) que
 *   supervisa el intercambio cíclico de datos entre el PLC maestro, buses de
 *   campo serie/Ethernet (Modbus RTU/TCP, CANopen, PROFINET) y nodos sensores
 *   inalámbricos (Wi-Fi MQTT, BLE, ZigBee, LoRaWAN, WirelessHART), validando:
 *     1. Contador de latido (Heartbeat) y secuencia de trama para detectar
 *        congelamiento o pérdida del enlace.
 *     2. Umbral de potencia recibida RSSI y relación señal a ruido SNR.
 *     3. Temporizador Watchdog de red (TON) de 100 ms con activación
 *        automática de estado seguro (Fail-Safe) en los actuadores de planta.
 * ============================================================================ *)

FUNCTION_BLOCK FB_GestorRedMecatronica_V04
VAR_INPUT
    (* Señales de Diagnóstico de Capa de Enlace y Transporte *)
    xTramaValidaCrcOk     : BOOL; (* 1 = Verificación polinómica CRC-16 correcta *)
    xParoEmergenciaRed_NC : BOOL; (* 1 = Canal de seguridad PROFIsafe / E-Stop OK *)
    byContadorHeartbeatRx : BYTE; (* Contador incremental enviado por el nodo remoto *)
    iRssiMedidoDbm        : INT;  (* Nivel de potencia recibida en dBm, ej. -72 *)
    iSnrMedidoDb          : INT;  (* Relación Señal a Ruido en dB, ej. -12 *)

    (* Datos de Proceso Recibidos en el Payload de la Trama *)
    iVariableProcesoRemota : INT; (* Valor crudo 0..27648 desde nodo remoto *)
    rConsignaLocalMaestro  : REAL := 75.0; (* Consigna a transmitir hacia actuador remoto *)
END_VAR

VAR_OUTPUT
    (* Banderas de Salud del Enlace y Conmutación Fail-Safe *)
    xEnlaceOperativoOk    : BOOL; (* 1 = Comunicación determinista activa *)
    xAlertaCalidadRfBaja  : BOOL; (* 1 = Margen de desvanecimiento reducido *)
    xFalloTimeoutFailSafe : BOOL; (* 1 = Pérdida de comunicación -> Parada segura *)

    (* Variables de Proceso Validadas hacia el Lazo de Control / SCADA *)
    rVariableProcesoSegura : REAL; (* Valor escalado o valor seguro por defecto *)
    iPalabraMandoTx        : INT;  (* Registro empaquetado para PDO / Modbus 0x06 *)
    byHeartbeatTx          : BYTE; (* Latido transmitido por el PLC maestro *)
END_VAR

VAR
    tonWatchdogRed         : TON;
    byHeartbeatPrevio      : BYTE := 16#00;
    xLatidoActualizado     : BOOL := FALSE;
    iUmbralMinimoRssiDbm   : INT  := -98;
END_VAR

(* ============================================================================
 * 1. DETECCIÓN DE ACTUALIZACIÓN DE LATIDO (HEARTBEAT) Y INTEGRIDAD CRC-16
 *    Equivalencia en Lista de Instrucciones (IL):
 *      LD     byContadorHeartbeatRx
 *      NE     byHeartbeatPrevio
 *      AND    xTramaValidaCrcOk
 *      ST     xLatidoActualizado
 * ============================================================================ *)
xLatidoActualizado := (byContadorHeartbeatRx <> byHeartbeatPrevio) AND xTramaValidaCrcOk;

IF xLatidoActualizado THEN
    byHeartbeatPrevio := byContadorHeartbeatRx;
END_IF;

(* ============================================================================
```

```
 * 2. TEMPORIZADOR WATCHDOG DE COMUNICACIÓN (TON DE 100 MS)
 *    Si no llegan tramas válidas dentro de la ventana determinista, activa Fail-Safe
 * ========================================================================== *)
tonWatchdogRed(
    IN := NOT xLatidoActualizado,
    PT := T#100MS
);

xFalloTimeoutFailSafe := tonWatchdogRed.Q OR (NOT xParoEmergenciaRed_NC);
xEnlaceOperativoOk    := (NOT xFalloTimeoutFailSafe) AND xTramaValidaCrcOk;

(* ==========================================================================
 * 3. SUPERVISIÓN DEL PRESUPUESTO DE ENLACE INALÁMBRICO (RSSI Y SNR)
 * ========================================================================== *)
IF (iRssiMedidoDbm < iUmbralMinimoRssiDbm) OR (iSnrMedidoDb < -15) THEN
    xAlertaCalidadRfBaja := TRUE;
ELSE
    xAlertaCalidadRfBaja := FALSE;
END_IF;

(* ==========================================================================
 * 4. GESTIÓN DETERMINISTA DE DATOS Y CONMUTACIÓN A ESTADO SEGURO (FAIL-SAFE)
 * ========================================================================== *)
IF xEnlaceOperativoOk THEN
    (* Escala la variable remota (0..27648 -> 0.0..100.0 %) y empaqueta consigna *)
    rVariableProcesoSegura := (INT_TO_REAL(iVariableProcesoRemota) / 27648.0) * 100.0;
    iPalabraMandoTx        := REAL_TO_INT((rConsignaLocalMaestro / 100.0) * 27648.0);
    byHeartbeatTx          := byHeartbeatTx + 16#01;
ELSE
    (* Estado Seguro (Fail-Safe): Fuerza consigna de potencia a cero *)
    rVariableProcesoSegura := 0.0;
    iPalabraMandoTx        := 0;
END_IF;

END_FUNCTION_BLOCK
```