Zephyr RTOS Programming – 3 Days (Multimedia & Debugging Focus)

Day 1 – Build System & Driver Fundamentals

1. **Zephyr Build System & Setup**
    - SDK install, first build/flash
    - CMake, Kconfig, Devicetree overview
    - Build outputs and artifacts
2. **Device Model and Drivers**
    - Zephyr device model: handles, init, APIs
    - GPIO mini-driver (LED)
    - UART send/receive (echo)
    - I²C/SPI read (sensor/EEPROM)
    - Direct Memory Access (DMA): mem → peripheral transfer demo
    - Platform driver comparison (Linux vs Zephyr)
3. **Multimedia Introduction**
    - Timer/counter as frame-counter demo
    - Zephyr vs Linux V4L2: conceptual comparison + demo

Day 2 – Execution Model, Threading & Kernel Services

1. **Execution Modes & Memory Protection**
    - User vs Kernel execution in Zephyr

- MPU protection and memory domains
- Access-control demo

2. **Multi-threading & Synchronization**
    - Thread creation with k_thread, priorities, preemption
    - Inter-thread communication: k_mutex, k_sem, k_msgq, spinlock
    - Stack size considerations
3. **Kernel Services & ISR Handling**
    - Dynamic memory allocation (k_malloc)
    - Message/event queues in multimedia scenarios
    - ISR handling: top/bottom halves, latency considerations
4. **Architecture Awareness**
    - RISC-V vs ARM in Zephyr (tooling, ports, BSP notes)

Day 3 – Debugging, Profiling & Multimedia Integration

1. **Debugging Applications**
    - printk/logging strategy
    - Core dump setup
    - GDB dumps, memory maps, register peek, backtraces
    - Spotting stack overflows, memfaults
2. **Crash Triage & Post-mortem Debugging**
    - Core dumps and analysis workflow
    - Real crash triage walkthrough
    - Kernel and user space debugging

    - Cross-boundary debug issues
3. **Profiling & Runtime Analysis**
    - TraceAnalyzer/SystemView for runtime profiling
    - Multimedia pipeline debug (timers + queues)
    - Monitoring memory usage, stack sizes, fragmentation
    - Measuring task/ISR duration with timing helpers
    - CPU utilization and memory analysis tools
4. **Multimedia Integration & Wrap-Up**
    - Clock control basics
    - LVGL bring-up (display + input drivers overview)
    - Event-driven input handling
    - Debugging checklists for real projects
    - Profiling + debugging in CI/testing flows
    - Recommended toolchain & workflow setup
    - Q&A + next steps

✅ This refined 3-day schedule has:

- **Day 1:** Fundamentals + drivers + multimedia basics
- **Day 2:** Execution model + threading + kernel services
- **Day 3:** Debugging, profiling, and multimedia integration with LVGL