RTOS & Multitasking

1. What is an RTOS?

- **RTOS (Real-Time Operating System):**
  A lightweight operating system that provides deterministic task scheduling for embedded systems.
- **Difference from Bare-Metal:**
  - Bare-metal → one big while(1) loop with interrupts.
  - RTOS → breaks application into multiple **tasks** with priorities.
- **Key Advantage:** Multitasking, modular design, predictable timing.

Use Cases: IoT gateways, robotics, motor control, medical devices, automotive systems.

2. RTOS Architecture

- **Kernel:** Core of RTOS (handles scheduling, task switching).
- **Tasks/Threads:** Independent execution units.
- **Scheduler:** Decides which task runs (based on priority & state).
- **System Tick (SysTick):** Hardware timer that triggers scheduler at fixed intervals.

Task States in FreeRTOS:

- Ready → waiting for CPU
- Running → currently executing
- Blocked → waiting for delay/event
- Suspended → stopped by developer

3. Task Management

- **Task Creation:** xTaskCreate()
- **Priorities:** Higher number = higher priority.
- **Context Switching:** Saving current task state, restoring next task.
- **Idle Task:** Always runs when no other task is ready.

Example (LED + UART tasks):

- Task1 → Blink LED every 1 sec
- Task2 → Print “Hello” on UART every 2 sec

4. Inter-Task Communication

RTOS provides safe ways for tasks/ISRs to communicate.

- **Semaphore:**
    - Binary → signal (e.g., Button ISR → Task).
    - Counting → manage multiple events.
- **Mutex (Mutual Exclusion):**
    - Protects shared resource (e.g., UART, SPI).
    - Prevents race conditions.
- **Queue:**
    - FIFO buffer between tasks.
    - Example: Task A (sensor) → Queue → Task B (UART).
- **Event Groups:**
    - Sync multiple tasks on one/more events.
    - Example: WiFi + Sensor ready → Send data.

5. Task Timing

- **Delays:**
    - vTaskDelay(ms) → wait relative time.
    - vTaskDelayUntil() → periodic precise scheduling.
- **Software Timers:**
    - Callback functions triggered after timeout.
    - Example: Send heartbeat message every 10s.

6. Memory Management

- Each task has its own **stack**.
- RTOS allocates stack & kernel objects from **heap**.
- Configurable in **FreeRTOSConfig.h**:
    - configTOTAL_HEAP_SIZE
    - configUSE_MUTEXES, configUSE_TIMERS

Common Pitfall: Stack overflow → must monitor task stack usage.

7. RTOS with Peripherals

- **ISR → Task Sync:**
  ISR signals semaphore → task handles logic.
  (Keep ISR short, do heavy work in tasks).
- **DMA + RTOS:**
  DMA interrupt signals task when buffer filled.
- **Low Power:**
  Idle task can put MCU into sleep mode.

8. Practical Exercises

1. LED + UART multitask demo.
2. Button ISR + Semaphore to task.
3. Queue for producer (sensor) and consumer (logger).
4. Periodic task (temperature read every 1s).
5. DMA buffer fill → task processing data.

9. Summary

- RTOS = multitasking + modular design.
- Core = tasks, priorities, scheduler.
- Use semaphores, mutexes, queues for safe communication.
- Essential for **industry-grade IoT & real-time systems**.

10. Program Flow Template

```
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"

SemaphoreHandle_t xSemaphore;

void vLEDTask(void *pvParameters) {
   while(1) {
      // Blink LED
      toggleLED();
      vTaskDelay(pdMS_TO_TICKS(1000));
   }
}

void vUARTTask(void *pvParameters) {
   while(1) {
      UART_Send("Hello\n");
      vTaskDelay(pdMS_TO_TICKS(2000));
   }
}

void vButtonTask(void *pvParameters) {
   while(1) {
      if(xSemaphoreTake(xSemaphore, portMAX_DELAY)) {
         toggleLED();
      }
   }
}
```

```
// ISR Example
void EXTI_ButtonISR(void) {
    BaseType_t xHigherPriorityTaskWoken = pdFALSE;
    xSemaphoreGiveFromISR(xSemaphore, &xHigherPriorityTaskWoken);
    portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}

int main(void) {
    hardware_init();
    xSemaphore = xSemaphoreCreateBinary();

    xTaskCreate(vLEDTask, "LED", 128, NULL, 1, NULL);
    xTaskCreate(vUARTTask, "UART", 128, NULL, 1, NULL);
    xTaskCreate(vButtonTask, "Button", 128, NULL, 2, NULL);

    vTaskStartScheduler();
    while(1);
}
```

11. References

- FreeRTOS Official Docs
- STM32 FreeRTOS Setup
- Microchip University FreeRTOS Course