

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)
SRM Nagar, Kattankulathur – 603 203.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

QUESTION BANK



III SEMESTER

M.E. – Communication Systems

PCU405 REAL TIME EMBEDDED SYSTEMS

Regulation – 2023

Academic Year 2026–2027 (ODD SEM)

Prepared by

Dr. C. Saravanakumar, Associate Professor/ECE



SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING QUESTION BANK

UNIT I – INTRODUCTION				
Real Time System – Embedded Systems – Architecture of Embedded System – Simple Programming for Embedded System – Process of Embedded System Development – Pervasive Computing – Information Access Devices – Smart Cards – Microcontrollers – ARM Processor -Real Time Microcontrollers.				
PART – A				
Q.No	Questions	CO	BT Level	Competence
1	Define a Real-Time System (RTS) according to R.J.A. Buhr.	CO1	BTL1	Remember
2	List four key hardware components that constitute the basic architecture of an embedded system.	CO1	BTL1	Remember
3	What is an Information Access Device (IAD)? Give two examples.	CO1	BTL1	Remember
4	Define Pervasive Computing (Ubiquitous Computing) in modern electronic environments.	CO1	BTL1	Remember
5	List the distinct operational phases involved in the process of embedded system development.	CO1	BTL1	Remember
6	State two fundamental structural differences between a standard microprocessor and a microcontroller.	CO1	BTL1	Remember
7	Name two primary execution modes supported by the ARM processor architecture.	CO1	BTL1	Remember
8	What is a Smart Card? State its major internal functional blocks.	CO1	BTL1	Remember
9	Define the term 'Cross-Compiler' as used in embedded system software development.	CO1	BTL1	Remember
10	What constitutes a Real-Time Microcontroller?	CO1	BTL1	Remember
11	State the purpose of the Program Counter (PC) and Link Register (LR) in an ARM processor register bank.	CO1	BTL1	Remember
12	Define deadline and response time in the context of real-time embedded execution.	CO1	BTL1	Remember
13	Distinguish between Hard Real-Time Systems and Soft Real-Time Systems with suitable communication examples.	CO1	BTL2	Understand
14	Write the significance of the Harvard architecture over Von Neumann architecture in real-time embedded processing.	CO1	BTL2	Understand
15	Mention the role of bit-manipulation operations in simple embedded C programming for I/O register configuration.	CO1	BTL2	Understand
16	Describe how a Smart Card maintains security during data transaction access.	CO1	BTL2	Understand
17	Write about the functionality of a Watchdog Timer (WDT) in maintaining the reliability of real-time microcontrollers.	CO1	BTL2	Understand
18	Compare the RISC architectural features of ARM processors with traditional CISC microcontrollers.	CO1	BTL2	Understand
19	What is the concept of 'Co-design' (Hardware/Software Co-design) in the development process of embedded systems.	CO1	BTL2	Understand
20	How did pervasive computing devices manage power constraints while remaining continuously available.	CO1	BTL2	Understand
21	Write the process of creating an executable image (Hex/Bin file) from C source code for an embedded target.	CO1	BTL2	Understand
22	Differentiate between Memory-Mapped I/O and Isolated (Port-Mapped) I/O in embedded system hardware design.	CO1	BTL2	Understand
23	How did interrupt latency impacts the temporal determinism of a real-time system.	CO1	BTL2	Understand
24	Illustrate the role of Information Access Devices in modern wireless communication networks.	CO1	BTL2	Understand
PART – B				
1	Construct a complete embedded C program to configure an 8-bit GPIO port of a real-time microcontroller to continuously acquire digital sensor data, apply bitwise masking operations (&, , ~, ^), and drive an actuator relay based on set thresholds.	CO1	BTL3	Apply
2	(a) Apply the concept of Hardware/Software Co-design to design an automated Smart Card Reader System used in secure access networks.	CO1	BTL3	Apply

	(b) Construct a block diagram illustrating its structural modules (processor, memory, smart card interface, display, and transceiver). (8+8)			
3	(a) An ARM processor-based embedded system must process periodic audio samples every 125 μ s using an Interrupt Service Routine (ISR). Construct the program execution flow and register saving/restoring mechanism during ISR entry and exit. (b) Demonstrate the application of ARM's Thumb instruction set mode in reducing memory footprint for this application. (8+8)	CO1	BTL3	Apply
4	Develop a complete end-to-end embedded software development lifecycle model for a real-time Wireless Sensor Node, detailing the practical deployment of In-Circuit Emulators (ICE), JTAG hardware debuggers, and Logic Analyzers during execution profiling.	CO1	BTL3	Apply
5	(a) Apply the principles of Pervasive Computing to design a Smart Health Monitoring wearable device acting as an Information Access Device (IAD). (b) Outline its system architecture including sensing, processing, low-power management, and wireless network communication modules. (8+8)	CO1	BTL3	Apply
6	Design a real-time control loop program in C using hardware timer interrupts to ensure deterministic periodic sampling and actuation in a closed-loop embedded temperature controller.	CO1	BTL3	Apply
7	(a) Design a Smart Card-based authentication system for cellular communication terminals. (b) Illustrate the step-by-step communication handshake, protocol interaction (ISO/IEC 7816), and memory mapping used in storing user credential data. (8+8)	CO1	BTL3	Apply
8	Design a real-time telemetry data processing unit using the ARM Cortex-M architecture, demonstrating the explicit configuration of the Nested Vectored Interrupt Controller (NVIC) and low-power modes to guarantee low-latency real-time response.	CO1	BTL3	Apply
9	Analyze the structural architecture of a Real-Time Embedded System by comparing bus-based and memory-centric topologies, evaluating how memory access latencies across RAM and Flash impact overall temporal determinism.	CO1	BTL4	Analyze
10	(a) Compare and analyze the architectural differences between an ARM processor (load-store architecture) and a conventional 8051/PIC microcontroller. (b) Deduce why 32-bit ARM architectures are preferred over 8-bit microcontrollers in high-throughput digital communication systems. (8+8)	CO1	BTL4	Analyze
11	Perform a detailed trade-off analysis (power, performance, execution determinism, and cost) among ASICs, FPGAs, and Real-Time Microcontrollers for deployment in Pervasive Computing hardware platforms.	CO1	BTL4	Analyze
12	(a) Break down the process of embedded software compilation and linking. Analyze the functional roles of cross-compilers, assemblers, linkers, and locators. (b) Examine the structure of a Map file and Linker Script file, detailing how code sections (.text, .data, .bss) are allocated to physical ROM/RAM memory addresses. (8+8)	CO1	BTL4	Analyze
13	Analyze the hardware and software architecture of an Information Access Device (IAD) in communication gateways, differentiating critically between real-time data path routing operations and control path management functions.	CO1	BTL4	Analyze
14	(a) Analyze the causes of non-deterministic delays in real-time microcontrollers (e.g., cache misses, pipeline stalls, interrupt nesting, and DMA bus contention). (b) Propose and justify architectural design strategies to mitigate these non-deterministic factors in safety-critical communication systems. (8+8)	CO1	BTL4	Analyze
15	Analyze the hardware and software design features that distinguish a standard microcontroller from a Real-Time Microcontroller, evaluating how hardware interrupt controllers and deterministic bus matrices guarantee strict execution timing.	CO1	BTL4	Analyze
16	(a) Analyze the execution pipeline stages of an ARM processor under interrupt handling conditions. (b) Critically examine how pipeline flushing and register banking affect the overall context switching time in time-critical communication workloads. (8+8)	CO1	BTL4	Analyze
17	Deconstruct the development process of an embedded system, analyzing how software unit testing, hardware-in-the-loop (HIL) simulation, and target deployment interact to minimize timing errors in production.	CO1	BTL4	Analyze

UNIT II – EMBEDDED/REAL TIME OPERATING SYSTEM

Operating System Concepts: Processes, Threads, Interrupts, Events - Real Time Scheduling Algorithms - Memory Management – Overview of Operating Systems for Embedded, Real Time Handheld Devices – Target Image Creation – Programming in Linux, Rtlinux, Vxworks, and Microcontroller Operating System Overview.

PART – A

Q.No	Questions	CO	BT Level	Competence
1	Define a process and a thread in the context of a Real-Time Operating System.	CO2	BTL 1	Remember
2	State the primary distinction between an Interrupt Service Routine (ISR) and a standard OS task.	CO2	BTL 1	Remember
3	What is Rate Monotonic Scheduling (RMS)?	CO2	BTL 1	Remember
4	Define Earliest Deadline First (EDF) scheduling algorithm.	CO2	BTL 1	Remember
5	What is meant by Priority Inversion in RTOS context?	CO2	BTL 1	Remember
6	List two common memory management schemes used in real-time embedded environments.	CO2	BTL 1	Remember
7	Define a 'Target Image' in embedded RTOS development.	CO2	BTL 1	Remember
8	What is RTLinux? How does it differ from standard Linux?	CO2	BTL 1	Remember
9	State two key architectural features of the VxWorks real-time operating system.	CO2	BTL 1	Remember
10	What is an event flag (or event group) in RTOS task synchronization?	CO2	BTL 1	Remember
11	Define the term 'Jitter' in real-time task scheduling.	CO2	BTL 1	Remember
12	List two prominent microcontroller-focused operating systems (e.g., FreeRTOS, μ C/OS-II).	CO2	BTL 1	Remember
13	Distinguish between preemptive and non-preemptive real-time scheduling paradigms.	CO2	BTL 2	Understand
14	Write about the concept of context switching time and its impact on RTOS execution overhead.	CO2	BTL 2	Understand
15	What is the working mechanism of the Priority Inheritance Protocol to mitigate priority inversion.	CO2	BTL 2	Understand
16	Why dynamic memory allocation (e.g., malloc/free) is generally avoided in hard real-time systems.	CO2	BTL 2	Understand
17	Mention the role of a Board Support Package (BSP) in generating a target OS image.	CO2	BTL 2	Understand
18	Compare the operational capabilities of Linux with RTLinux for real-time communication tasks.	CO2	BTL 2	Understand
19	How did interrupt latency and interrupt response time are measured in an RTOS kernel.	CO2	BTL 2	Understand
20	Write the process of cross-compiling and linking an RTOS target image for an embedded platform.	CO2	BTL 2	Understand
21	What are the memory protection mechanisms used in handheld device operating systems (e.g., Android, Linux).	CO2	BTL 2	Understand
22	What is the interaction between software event triggers and hardware interrupt signals in an RTOS.	CO2	BTL 2	Understand
23	Write the CPU utilization bound formula for Rate Monotonic Scheduling.	CO2	BTL 2	Understand
24	Differentiate microkernel architectures from monolithic kernel architectures in embedded systems.	CO2	BTL 2	Understand

PART – B

1	A real-time communication terminal executes three periodic tasks: T1 (P1=4ms, C1=1ms), T2 (P2=8ms, C2=2ms), and T3 (P3=12ms, C3=3ms). Apply the Rate Monotonic Scheduling (RMS) algorithm to construct the execution timeline chart over a 24ms hyperperiod, verify schedulability using Liu and Layland's utilization bound, and analyze the feasibility.	CO2	BTL 3	Apply
2	(a) Apply the Earliest Deadline First (EDF) dynamic scheduling algorithm to schedule a set of periodic tasks in a communication gateway. (b) Demonstrate through time-line construction how EDF achieves up to 100% processor utilization compared to fixed-priority scheduling. (8+8)	CO2	BTL 3	Apply
3	(a) Construct a real-time POSIX multithreaded application snippet in RTLinux to process periodic audio data streams. (b) Demonstrate how mutexes and semaphores are used to prevent race conditions during	CO2	BTL 3	Apply

	real-time inter-thread data sharing. (8+8)			
4	Design a complete target OS image creation workflow for an embedded Linux gateway, detailing bootloader configuration (U-Boot), kernel compilation, device tree setup, root filesystem construction, and final target image deployment.	CO2	BTL 3	Apply
5	(a) Apply the Priority Ceiling Protocol (PCP) to a multi-task real-time system experiencing severe priority inversion. (b) Illustrate with timing diagrams how PCP prevents both unbounded priority inversion and mutual deadlocks. (8+8)	CO2	BTL 3	Apply
6	Develop a real-time event-driven application architecture in VxWorks OS for an automated telemetry receiver, showing the deployment of task spawns, message queues, and semaphores for low-latency interrupt-to-task signaling.	CO2	BTL 3	Apply
7	(a) Apply fixed-block memory pool management techniques to replace standard dynamic memory allocation in a real-time embedded system. (b) Write an embedded C implementation of a deterministic fixed-size memory pool allocator guaranteeing O(1) allocation time. (8+8)	CO2	BTL 3	Apply
8	Construct a complete software design for a Microcontroller OS (e.g., FreeRTOS) managing three concurrent tasks: sensor sampling, data processing, and wireless packet transmission, demonstrating priority assignment and event synchronization.	CO2	BTL 3	Apply
9	Analyze the architectural differences between dual-kernel real-time operating systems (RTLinux) and native microkernel RTOSs (VxWorks, QNX), evaluating how each handles interrupt dispatching, kernel preemptibility, and context switching latencies.	CO2	BTL 4	Analyze
10	(a) Compare and analyze static priority (RMS) versus dynamic priority (EDF) real-time scheduling algorithms. (b) Evaluate their performance under runtime task overload conditions and run-time processing overheads in embedded systems. (8+8)	CO2	BTL 4	Analyze
11	Analyze the memory management requirements for handheld embedded devices (e.g., mobile terminals), evaluating virtual memory vs flat memory architectures and the trade-offs between dynamic memory fragmentation and execution determinism.	CO2	BTL 4	Analyze
12	(a) Analyze the internal architecture of a Microcontroller Operating System (e.g., μ C/OS-II or FreeRTOS). (b) Examine the task control block (TCB) structure, tick timer implementation, and low-overhead context switching routines during task scheduling. (8+8)	CO2	BTL 4	Analyze
13	Deconstruct the target image generation pipeline for an embedded RTOS, analyzing how configuration scripts, linker locators, memory mapping, and target initialization code interact to produce an executable flash image.	CO2	BTL 4	Analyze
14	(a) Analyze the execution path from a hardware interrupt signal assertion to the completion of a high-priority user-level task in a real-time kernel. (b) Identify and mathematically express all latency components including interrupt latency, ISR execution time, queuing delay, and scheduling context switch time. (8+8)	CO2	BTL 4	Analyze
15	Analyze the operational trade-offs of using general-purpose embedded Linux versus real-time operating systems (VxWorks / RTLinux) in high-throughput digital communication systems, focusing on determinism, driver availability, and resource utilization.	CO2	BTL 4	Analyze
16	(a) Analyze the priority inversion problem using a three-task scenario (High, Medium, Low priority) sharing a single resource via semaphores. (b) Evaluate and compare the solutions offered by Priority Inheritance Protocol (PIP) and Priority Ceiling Protocol (PCP). (8+8)	CO2	BTL 4	Analyze
17	Analyze the process management and thread synchronization primitives in VxWorks versus RTLinux, evaluating how each framework ensures time-deterministic execution in safety-critical communication applications.	CO2	BTL 4	Analyze

UNIT III – CONNECTIVITY				
Wireless Connectivity - Bluetooth – Other Short-Range Protocols – Wireless Application Environment – Service Discovery – Middleware.				
PART – A				
Q.No	Questions	CO	BT Level	Competence
1	Define a Bluetooth Piconet and Scatternet.	CO3	BTL 1	Remember
2	List the core layers of the Bluetooth protocol stack.	CO3	BTL 1	Remember
3	What is ZigBee (IEEE 802.15.4) and state its primary operating frequency bands?	CO3	BTL 1	Remember
4	Define Near Field Communication (NFC) and state its maximum operating distance.	CO3	BTL 1	Remember

5	What is the primary function of the Service Discovery Protocol (SDP) in Bluetooth?	CO3	BTL 1	Remember
6	Define Wireless Application Environment (WAE) in mobile/embedded communication.	CO3	BTL 1	Remember
7	What is middleware in the context of real-time embedded networks?	CO3	BTL 1	Remember
8	List two prominent short-range wireless protocols besides Bluetooth.	CO3	BTL 1	Remember
9	What is Universal Plug and Play (UPnP) service discovery?	CO3	BTL 1	Remember
10	Define Ultra-Wideband (UWB) communication technology.	CO3	BTL 1	Remember
11	State the purpose of the Logical Link Control and Adaptation Protocol (L2CAP) in Bluetooth.	CO3	BTL 1	Remember
12	What is a Wireless Access Point (WAP) profile in WAE environments?	CO3	BTL 1	Remember
13	Distinguish between Bluetooth Low Energy (BLE) and Classic Bluetooth in terms of power consumption and data throughput.	CO3	BTL 2	Understand
14	Write about the frequency-hopping spread spectrum (FHSS) mechanism used in Bluetooth hardware interfaces.	CO3	BTL 2	Understand
15	What is the dynamic service registration and discovery mechanism in embedded wireless networks.	CO3	BTL 2	Understand
16	Compare ZigBee and Z-Wave technologies for short-range home automation environments.	CO3	BTL 2	Understand
17	Mention the role of message-oriented middleware (MOM) in real-time sensor networks.	CO3	BTL 2	Understand
18	How did the Wireless Application Protocol (WAP) stack maps onto standard ISO/OSI layer model.	CO3	BTL 2	Understand
19	Describe the security challenge during the pairing process in Bluetooth communication.	CO3	BTL 2	Understand
20	What are the operational modes of NFC (Peer-to-Peer, Reader/Writer, Card Emulation).	CO3	BTL 2	Understand
21	How did object-oriented middleware (e.g., CORBA/eCORBA) abstracts underlying hardware heterogeneity.	CO3	BTL 2	Understand
22	Write the significance of publish-subscribe middleware protocols (e.g., MQTT) in real-time embedded systems.	CO3	BTL 2	Understand
23	The Bluetooth manages interference with Wi-Fi (IEEE 802.11) operating in the same 2.4 GHz ISM band. How?	CO3	BTL 2	Understand
24	Differentiate dynamic service discovery protocols: SDP vs. mDNS/DNS-SD.	CO3	BTL 2	Understand

PART – B

1	Design a complete wireless sensor node network using the Bluetooth Low Energy (BLE) protocol stack. Construct the GATT (Generic Attribute Profile) service architecture, including custom profiles, characteristics, and descriptors to acquire telemetry data deterministically.	CO3	BTL 3	Apply
2	(a) Apply the IEEE 802.15.4 / ZigBee protocol stack to establish a wireless mesh network topology for industrial process control. (b) Demonstrate how beacon-enabled slotted CSMA/CA mechanisms guarantee real-time latency bounds for critical control messages. (8+8)	CO3	BTL 3	Apply
3	(a) Construct a dynamic Service Discovery Protocol (SDP) transaction sequence in a mobile medical health monitoring system. (b) Illustrate the step-by-step query/response handshake between a wearable sensor client and a gateway host using SDP attributes. (8+8)	CO3	BTL 3	Apply
4	Develop a lightweight publish-subscribe embedded middleware architecture (e.g., MQTT/CoAP over UDP) for constrained wireless access nodes, showing how real-time QoS levels are implemented to guarantee reliable packet delivery.	CO3	BTL 3	Apply
5	(a) Apply Near Field Communication (NFC) protocol principles to design a contactless smart access control terminal. (b) Construct the complete signal-level authentication sequence and APDU command exchange between the card reader and target device. (8+8)	CO3	BTL 3	Apply
6	Design a Wireless Application Environment (WAE) software framework for an embedded mobile terminal, detailing the interaction among WAP gateway, WML user interface, and underlying transport layer protocols.	CO3	BTL 3	Apply
7	(a) Apply Ultra-Wideband (UWB) short-range wireless technology to construct a real-time indoor localization and distance estimation system. (b) Demonstrate mathematically how Two-Way Ranging (TWR) algorithms calculate precise time-of-flight (ToF) under multipath fading. (8+8)	CO3	BTL 3	Apply
8	Construct a hybrid wireless gateway integrating Bluetooth LE, ZigBee, and Wi-Fi interfaces, demonstrating the practical application of a unified embedded middleware layer to seamlessly route cross-protocol sensor packets in real time.	CO3	BTL 3	Apply
9	Analyze the structural architecture of the Bluetooth protocol stack (Baseband, LMP, L2CAP, SDP, RFCOMM), critically evaluating how timing synchronization, link control, and channel multiplexing are managed to maintain hard real-time guarantees.	CO3	BTL 4	Analyze

10	(a) Compare and analyze short-range wireless protocols: Bluetooth 5.0, ZigBee (802.15.4), Z-Wave, and UWB. (b) Evaluate their performance in terms of throughput, latency, energy dissipation, network capacity, and co-existence capability in congested ISM spectrums. (8+8)	CO3	BTL 4	Analyze
11	Deconstruct the dynamic service discovery mechanisms across heterogeneous wireless networks, evaluating the performance trade-offs among Bluetooth SDP, UPnP (SSDP), and mDNS/DNS-SD in terms of network overhead, discovery latency, and memory footprint.	CO3	BTL 4	Analyze
12	(a) Analyze the role of embedded middleware in distributed real-time systems. (b) Evaluate object-oriented middleware (RT-CORBA) vs message-oriented middleware (MQTT-SN / DDS) for low-latency communication in resource-constrained hardware. (8+8)	CO3	BTL 4	Analyze
13	Analyze the security vulnerabilities inherent in short-range wireless connectivity (e.g., eavesdropping, man-in-the-middle attacks, replay attacks in Bluetooth/ZigBee), and evaluate hardware/software cryptographic mitigation techniques.	CO3	BTL 4	Analyze
14	(a) Analyze the structural architecture of the Wireless Application Environment (WAE) and WAP stack. (b) Evaluate how gateway proxy conversion mitigates bandwidth constraints and latency bottlenecks over wireless cellular channels. (8+8)	CO3	BTL 4	Analyze
15	Analyze the co-existence challenges of multiple short-range wireless protocols (Bluetooth, ZigBee, Wi-Fi) operating simultaneously in the 2.4 GHz ISM band, evaluating adaptive frequency hopping (AFH) and clear channel assessment (CCA) mitigation techniques.	CO3	BTL 4	Analyze
16	(a) Analyze the power management states of Bluetooth Low Energy (Advertising, Scanning, Initiating, Connection states). (b) Deduce mathematical models for energy consumption as a function of connection interval and slave latency in long-term embedded sensor deployments. (8+8)	CO3	BTL 4	Analyze
17	Deconstruct the end-to-end communication path of an IoT embedded gateway, analyzing how short-range wireless packets are received, decoded by protocol-specific drivers, translated by middleware, and dispatched over long-range backhaul networks under strict real-time constraints.	CO3	BTL 4	Analyze

UNIT IV – REAL TIME UML

The Rapid Object-Oriented Process for Embedded Systems (ROPES) Process. MDA and Platform Independent Models- Scheduling Model-Based Projects- Model Organization Principles- Working with Model-Based Projects - Object Orientation with UML 2.0-Structural Aspects-Object Orientation with UML 2.0-Dynamic Aspects-UML Profile for Schedulability, Performance, and Time.

PART – A

Q.No	Questions	CO	BT Level	Competence
1	What is the ROPES (Rapid Object-Oriented Process for Embedded Systems) process?	CO4	BTL 1	Remember
2	Define Model-Driven Architecture (MDA).	CO4	BTL 1	Remember
3	What is a Platform Independent Model (PIM) in MDA?	CO4	BTL 1	Remember
4	What is a Platform Specific Model (PSM)?	CO4	BTL 1	Remember
5	List two key structural diagrams used in UML 2.0 for real-time systems modeling.	CO4	BTL 1	Remember
6	Name two dynamic behavioral diagrams in UML 2.0 used for specifying execution sequences.	CO4	BTL 1	Remember
7	Define a UML Profile.	CO4	BTL 1	Remember
8	What is the UML Profile for Schedulability, Performance, and Time (SPT)?	CO4	BTL 1	Remember
9	State the primary objective of scheduling model-based projects in embedded development.	CO4	BTL 1	Remember
10	Define 'Stereotype' and 'Tagged Value' in UML extension mechanisms.	CO4	BTL 1	Remember
11	What constitutes a 'Composite Structure Diagram' in UML 2.0?	CO4	BTL 1	Remember
12	List the primary phases involved in the ROPES development lifecycle.	CO4	BTL 1	Remember
13	Distinguish between Platform Independent Models (PIM) and Platform Specific Models (PSM) with an embedded system example.	CO4	BTL 2	Understand
14	What is the role of Model Transformation in Model-Driven Architecture (MDA).	CO4	BTL 2	Understand
15	How did active classes and ports are used to model concurrency in UML 2.0 structural diagrams.	CO4	BTL 2	Understand
16	Write the dynamic behavioral execution of a UML Statechart Diagram for real-time state transitions.	CO4	BTL 2	Understand

17	State the key model organization principles for managing large-scale model-based projects using UML packages.	CO4	BTL 2	Understand
18	How time constraints and deadlines are modeled using the UML SPT profile.	CO4	BTL 2	Understand
19	Compare the structural representation of a class diagram with a component diagram in UML 2.0.	CO4	BTL 2	Understand
20	Write the iterative macro-process and micro-process loops within the ROPES methodology.	CO4	BTL 2	Understand
21	What is the mechanism of message passing in UML 2.0 Sequence and Communication diagrams.	CO4	BTL 2	Understand
22	How schedulability analysis annotations are embedded into UML models using SPT stereotypes.	CO4	BTL 2	Understand
23	What are the practical challenges and mitigation strategies when working with collaborative model-based engineering projects.	CO4	BTL 2	Understand
24	Differentiate between structural aspects and dynamic aspects of object-oriented modeling in UML 2.0.	CO4	BTL 2	Understand
PART – B				
1	Apply the ROPES (Rapid Object-Oriented Process for Embedded Systems) methodology to model an automated real-time telemetry node. Construct the system lifecycle workflow across Analysis, Architecture, Design, Implementation, and Testing phases.	CO4	BTL 3	Apply
2	(a) Apply the Model-Driven Architecture (MDA) framework to map a Platform Independent Model (PIM) of a communication router into an ARM Cortex-M Platform Specific Model (PSM). (b) Demonstrate the model transformation steps and automated code generation mechanisms. (8+8)	CO4	BTL 3	Apply
3	(a) Construct a UML 2.0 Composite Structure Diagram and Class Diagram to model the structural aspects of a real-time smart card reader unit. (b) Demonstrate the application of active objects, ports, provided interfaces, and required interfaces for low-latency hardware abstraction. (8+8)	CO4	BTL 3	Apply
4	Develop a comprehensive dynamic behavioral model for a real-time packet processing engine using UML 2.0 Sequence Diagrams and Statechart Diagrams, showing concurrent execution, guard conditions, and interrupt-driven state transitions.	CO4	BTL 3	Apply
5	(a) Apply the UML Profile for Schedulability, Performance, and Time (SPT) to annotate a real-time task model with periodicity, execution time, and hard deadlines. (b) Demonstrate how these annotations are used by schedulability analysis tools to prove feasibility. (8+8)	CO4	BTL 3	Apply
6	Apply model organization principles to design a multi-package UML model for a distributed wireless sensor gateway, detailing package dependencies, visibility, model management, and configuration control across engineering teams.	CO4	BTL 3	Apply
7	(a) Apply UML 2.0 Activity and Timing diagrams to model the real-time execution bounds of a wireless transceiver pipeline. (b) Demonstrate how timing constraints and latency budgets are explicitly validated on the model timeline. (8+8)	CO4	BTL 3	Apply
8	Construct a complete model-based project scheduling framework for an embedded software development project, demonstrating project decomposition, milestone verification using executable models, and iterative increment planning under ROPES.	CO4	BTL 3	Apply
9	Analyze the structural aspects of object orientation in UML 2.0 for real-time systems, critically evaluating how composite structures, active classes, ports, and connectors improve modularity and execution determinism compared to traditional object models.	CO4	BTL 4	Analyze
10	(a) Compare and analyze the dynamic aspects of UML 2.0 by contrasting Statechart Diagrams, Sequence Diagrams, and Activity Diagrams. (b) Evaluate which dynamic model is best suited for specifying real-time concurrency, asynchronous event handling, and interrupt processing. (8+8)	CO4	BTL 4	Analyze
11	Analyze the MDA transformation paradigm (PIM to PSM to Code) in safety-critical real-time embedded systems, evaluating the benefits, limitations, toolchain constraints, and code efficiency trade-offs.	CO4	BTL 4	Analyze
12	(a) Analyze the domain architecture of the UML Profile for Schedulability, Performance, and Time (SPT). (b) Examine how SPT stereotypes (`<<SAEngine>>`, `<<SATask>>`, `<<SAResource>>`) express priority inversion, resource sharing, and response time metrics. (8+8)	CO4	BTL 4	Analyze
13	Deconstruct the ROPES process lifecycle, analyzing how iterative spiraling, continuous target integration, and model-based testing eliminate system integration risks early in embedded development.	CO4	BTL 4	Analyze

14	(a) Analyze the organizational principles for large model-based embedded software projects. (b) Critically evaluate package partitioning strategies (by sub-system, by layer, by feature) and model version control challenges in collaborative team environments. (8+8)	CO4	BTL 4	Analyze
15	Analyze the execution semantics of UML 2.0 active objects and message dispatching mechanisms, evaluating run-to-completion (RTC) semantics versus multi-threaded execution in time-critical communication workloads.	CO4	BTL 4	Analyze
16	(a) Analyze how schedulability models integrate with dynamic behavioral models in UML 2.0. (b) Evaluate how changes in task execution time (C _i) or period (P _i) in the SPT profile dynamically impact the underlying sequence and timing models. (8+8)	CO4	BTL 4	Analyze
17	Deconstruct a complex model-driven embedded engineering workflow, analyzing how executable models (Model-in-the-Loop, Software-in-the-Loop, Processor-in-the-Loop) accelerate verification compared to traditional document-centric methodologies.	CO4	BTL 4	Analyze

UNIT V – SOFTWARE DEVELOPMENT AND APPLICATION

Concurrency – Exceptions – Tools – Debugging Techniques – Optimization –Interfacing Digital Camera with USB Port. Interfacing of Sensors and Actuators for a Real Time Industrial Application.

PART – A

Q.No	Questions	CO	BT Level	Competence
1	Define Concurrency in real-time embedded systems.	CO5	BTL 1	Remember
2	What is an Exception in an embedded processor?	CO5	BTL 1	Remember
3	List two hardware tools used for debugging embedded software.	CO5	BTL 1	Remember
4	What is an In-Circuit Emulator (ICE)?	CO5	BTL 1	Remember
5	Define Code Optimization in real-time programming.	CO5	BTL 1	Remember
6	State the role of the USB Isochronous Transfer Mode when interfacing a digital camera.	CO5	BTL 1	Remember
7	Define Sensor Signal Conditioning.	CO5	BTL 1	Remember
8	What is an Actuator driver circuit?	CO5	BTL 1	Remember
9	List two common software debugging techniques for real-time systems.	CO5	BTL 1	Remember
10	Define Race Condition in a concurrent embedded application.	CO5	BTL 1	Remember
11	State the primary function of a USB Video Class (UVC) driver.	CO5	BTL 1	Remember
12	What is a Watchdog Timer (WDT) and its role in exception handling?	CO5	BTL 1	Remember
13	Distinguish between preemptive multitasking and cooperative multitasking in concurrent execution.	CO5	BTL 2	Understand
14	State about the exception propagation and handling lifecycle in a real-time OS (RTOS).	CO5	BTL 2	Understand
15	How JTAG (Joint Test Action Group) debugging works at the hardware boundary level.	CO5	BTL 2	Understand
16	Mention the difference between loop unrolling and inline expansion for memory/speed optimization.	CO5	BTL 2	Understand
17	Write about the end-to-end data transfer flow, when interfacing a USB digital camera with a micro-controller.	CO5	BTL 2	Understand
18	What is the working principle of industrial optical isolators in sensor interfacing.	CO5	BTL 2	Understand
19	Compare hardware breakpoints versus software breakpoints during debugging.	CO5	BTL 2	Understand
20	How did priority inversion can occur in concurrent real-time systems and state its solution.	CO5	BTL 2	Understand
21	What is the role of Pulse Width Modulation (PWM) in driving industrial actuators like DC motors.	CO5	BTL 2	Understand
22	How did memory footprint optimization impacts execution speed in resource-constrained MCUs.	CO5	BTL 2	Understand
23	Write the mechanism of interrupt service routine (ISR) latency in real-time exception processing.	CO5	BTL 2	Understand
24	Differentiate between analog sensor interfacing (ADC conversion) and digital sensor interfacing (I2C/SPI).	CO5	BTL 2	Understand

PART – B

1	Design a concurrent real-time control system for an industrial assembly line. Implement synchronization mechanisms (semaphores, mutexes, message queues) to prevent race conditions and ensure thread safety across concurrent sensor sampling tasks.	CO5	BTL 3	Apply
2	(a) Apply structured exception handling techniques to handle hardware faults, memory management unit (MMU) traps, and arithmetic overflows in a safety-critical embedded system. (b) Demonstrate how a fail-safe backup state is safely achieved upon exception detection.	CO5	BTL 3	Apply

	(8+8)			
3	(a) Apply JTAG and Logic Analyzer debugging tools to diagnose a sporadic deadlock condition in a multi-threaded system. (b) Illustrate the step-by-step trace capture and signal analysis procedure. (8+8)	CO5	BTL 3	Apply
4	Develop a real-time memory and execution optimization strategy for a resource-constrained microcontroller, applying compiler optimization flags, inline assembly, register allocation, and memory-mapped IO structuring to achieve maximum throughput.	CO5	BTL 3	Apply
5	(a) Apply USB protocol primitives to interface a CMOS digital camera with an embedded host processor via a USB port. (b) Construct the USB descriptor setup, endpoint configuration, and DMA buffer allocation for real-time video streaming. (8+8)	CO5	BTL 3	Apply
6	Design a complete industrial temperature and pressure monitoring system, detailing sensor signal conditioning (instrumentation amplifiers, active filters, ADC), actuator interface circuits (relays, solenoids, PWM drivers), and noise immunity techniques.	CO5	BTL 3	Apply
7	(a) Apply advanced profiling tools (e.g., gprof, ETM trace) to identify timing bottlenecks in an embedded image processing algorithm. (b) Demonstrate code restructuring techniques that achieve a 40% reduction in Worst-Case Execution Time (WCET). (8+8)	CO5	BTL 3	Apply
8	Construct an end-to-end real-time industrial robotics actuator controller interface integrating optical encoders, DC servo motors, H-bridge driver circuits, and PID control algorithms implemented within an RTOS task structure.	CO5	BTL 3	Apply
9	Analyze the concurrency issues in multi-tasking real-time systems, critically evaluating priority inversion, deadlock, and livelock scenarios. Evaluate the performance of Priority Inheritance Protocol (PIP) versus Priority Ceiling Protocol (PCP) in mitigating these risks.	CO5	BTL 4	Analyze
10	(a) Compare and analyze hardware-assisted debugging tools (In-Circuit Emulators, JTAG/SWD, Logic Analyzers) versus software-based debugging techniques (logging, assertions, software break points). (b) Evaluate their impact on target system execution real-time determinism (probe effect). (8+8)	CO5	BTL 4	Analyze
11	Deconstruct the USB communication stack architecture for high-bandwidth embedded peripherals, analyzing how control, bulk, interrupt, and isochronous transfer modes perform under real-time image data transfer constraints from a USB digital camera.	CO5	BTL 4	Analyze
12	(a) Analyze compiler optimization techniques (loop unrolling, strength reduction, dead code elimination) versus manual assembly-level tuning for real-time microcontrollers. (b) Evaluate the trade-offs between execution speed, code size, and RAM utilization. (8+8)	CO5	BTL 4	Analyze
13	Analyze the complete signal path and exception handling workflow of an industrial automated plant interface, evaluating noise susceptibility, ground loop errors, ADC quantisation errors, and system response latency under fault conditions.	CO5	BTL 4	Analyze
14	(a) Analyze the execution overhead of nested interrupt processing and exception vectors in real-time processors. (b) Deduce mathematical expressions for total interrupt latency and evaluate its impact on task deadline compliance. (8+8)	CO5	BTL 4	Analyze
15	Analyze the design considerations for interfacing smart industrial sensors (I2C, SPI, RS-485 Modbus) and high-power actuators (stepper motors, pneumatic valves), evaluating electrical isolation, thermal management, and EMI/EMC compliance.	CO5	BTL 4	Analyze
16	(a) Analyze static code analysis vs dynamic memory profiling in embedded software verification. (b) Evaluate how memory leaks, stack overflow, and pointer corruption are detected and mitigated in real-time application code. (8+8)	CO5	BTL 4	Analyze
17	Deconstruct a real-time vision-guided industrial pick-and-place system, analyzing how USB camera image acquisition, embedded feature extraction, concurrency handling, and actuator motion triggering are integrated under tight real-time latency bounds.	CO5	BTL 4	Analyze