For Exam Preparation (Don't limit only on this)

Unit I: Introduction to Embedded Systems (3 hrs.)

1.1 Definition and Overview of Embedded Systems
1.2 Embedded Systems vs. General-Purpose Systems
1.3 Applications and Domains (Automotive, Healthcare, Industrial Control, IoT, etc.)
1.4 Key Components: Microcontrollers, Sensors, Actuators, Peripherals

(12-15 marks) (Maybe 2 long=15 marks or 1 long and 1 short note =12 marks)

Important Topics and Questions

1. Essential features of ES.
2. What is an ES? Why is it so hard to define? Explain the different types of embedded systems.
3. Differences between ES and General-purpose systems.
4. How ES is different from conventional computing?
5. Components and Applications of ES.
6. Domain of ES (Focus on -Automotive, Healthcare, Industrial Control, IoT).
7. Explain different components of ES with a figure. Prove that (**ATM, Washing Machines, ABS, CCTV Camera, and automatic fuel machines …….** (**1 example**)) is an example of ES.
8. You are tasked with designing an embedded system (Home security system, Healthcare, Weather, Education, automotive, Agriculture……. (**1 example**)) system that includes **any features (For Smart home-**motion detection, door sensors, and alarms.) Discuss the design metrics, constraints, structure, challenges, and steps in the design process for developing this system.
9. Classify ES Based on Performance and Functional Requirements, Deterministic behavior and triggering.
10. Explain all the design steps and processes to design a temperature control system and GPS in detail.
11. List and define the three main characteristics of ES that distinguish ES from other computing systems.
12. Define

    a. **Microcontrollers**
    b. **Sensors**
    c. **Actuators**
    d. **Peripherals**

Unit II: Programming for Embedded Systems (5 hrs.)

2.1 Overview of AVR Architecture
2.2 Embedded C Programming for Microcontroller:
- Introduction to C for Embedded Systems.
- Data Types, Control Structures, and Pointers

- Memory Management
- AVR Interrupt Handling
- Input and output ports interfacing on AVR
- Timers and Counters in AVR
- Serial Communication in AVR

(15-22 marks) (Maybe 3 long=22 marks or 2 long =15 marks or 2 long and 1 short notes=20)

1. Explain the key features of the AVR(Atmega16/32) architecture with a figure and also include the role of General-Purpose Registers, ALU, Program Counter, and Watchdog Timer in AVR.
2. How I/O ports are configured in AVR microcontrollers? Explain with an example.
3. What is RISC architecture, and how does it benefit AVR microcontrollers?
4. Explain the role of the following AVR components: SRAM, EEPROM, and Flash memory.
5. How is data transferred between the CPU and peripherals in an AVR microcontroller?
6. Explain the different types of memory available in AVR microcontrollers.
7. Write a program to allocate memory dynamically and free it in Embedded C.
8. What is an interrupt? How does an AVR microcontroller handle interrupts?
9. Explain the steps involved in configuring and enabling an interrupt in AVR.
10. What is the difference between digital and analog inputs in AVR? Provide an example of each
11. What is the difference between PORTx, PINx, and DDRx registers in AVR?
12. Explain the different modes of operation of timers in AVR. Differentiate between Timer 0, Timer 1, and Timer 2 of AVR microcontroller.
13. What is serial communication? How is it implemented in AVR microcontrollers? Also mention synchronous and asynchronous serial communication
14. If numerical

Example 9-1

Find the value for TCCR0 if we want to program Timer0 in Normal mode, no prescaler. Use AVR's crystal oscillator for the clock source.

Solution:

TCCR0 =

0	0	0	0	0	0	0	1
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Example 9-2

Find the timer's clock frequency and its period for various AVR-based systems, with the following crystal frequencies. Assume that no prescaler is used.
(a) 10 MHz (b) 8 MHz (c) 1 MHz

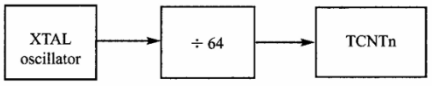
Solution:

(a) F = 10 MHz and T = 1/10 MHz = 0.1 µs
(b) F = 8 MHz and T = 1/8 MHz = 0.125 µs
(c) F = 1 MHz and T = 1/1 MHz = 1 µs

Example 9-10

Find the timer's clock frequency and its period for various AVR-based systems, with the following crystal frequencies. Assume that a prescaler of 1:64 is used.
(a) 8 MHz (b) 16 MHz (c) 10 MHz

Solution:



(a) 1/64 × 8 MHz = 125 kHz due to 1:64 prescaler and T = 1/125 kHz = 8 µs
(b) 1/64 × 16 MHz = 250 kHz due to prescaler and T = 1/250 kHz = 4 µs
(c) 1/64 × 10 MHz = 156.2 kHz due to prescaler and T = 1/156 kHz = 6.4 µs

Example 9-11

Find the value for TCCR0 if we want to program Timer0 in Normal mode with a prescaler of 64 using internal clock for the clock source.

Solution:

From Figure 9-5 we have TCCR0 = 0000 0011; XTAL clock source, prescaler of 64.

TCCR0 =

0	0	0	0	0	0	1	1
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

15. What is the role of prescaler in AVR timers? How do you configure them?
16. Explain the differences between standard C programming and Embedded C programming. Why is Embedded C preferred for microcontroller programming? Provide examples.
17. Discuss the role of header files in Embedded C programming with examples. Explain the use of control structures (if-else, switch-case) in Embedded C programming with examples.

Program

1. Write an AVR program in C to generate a square wave using Timer 0 and Timer 1 AVR microcontroller.
2. Write a program to store and retrieve a value from the EEPROM memory of an AVR microcontroller.
3. Discuss how memory is managed in Embedded C for AVR microcontrollers. Explain with C code.
4. Write an AVR program in C to toggle a bit of a specific port using bitwise operation.
5. Write a simple Embedded C program to blink an LED connected to PORTB pin 0 of an AVR microcontroller.
6. Write a program to turn on an LED when a button connected to PD2 is pressed.
7. Write an AVR program in C to read a digital input from a switch and turn on an LED.
8. Write a program to demonstrate the use of an external interrupt on an AVR microcontroller.

Unit III: Real-Time Operating Systems (RTOS)(5 hrs.)

3.1 Concepts of Real-Time Systems
3.2 Introduction to RTOS (e.g., FreeRTOS, VxWorks)
3.3 Task Scheduling, Context Switching, Task Synchronization
3.4 Resource Sharing, Deadlock, and Priority Inversion
3.5 Multithreading and Multi-tasking in RTOS

(15-20 marks)(May be 2 long=15 marks or 2 long and 1 short notes=20 marks)

1. What is RTOS and how it is beneficial in ES applications? Explain with block diagram
2. Explain the conditions that can lead to deadlock in RTOS-based ES.
3. Discuss semaphore-based and mutex-based resource-sharing techniques in RTOS.
4. How does RTOS differ from a general-purpose operating system (GPOS)in an Embedded system explain with key features of an RTOS. Also, differentiate between FreeRTOS and VxWorks.

5. What does a real-time embedded system mean? How does RTOS ensure fairness in scheduling? Discuss trade-offs between fairness, latency, and complexity in task synchronization.
6. If two tasks are deadlocked over a resource, how does an RTOS detect and resolve the deadlock? How would you handle synchronization between tasks and interrupts in a time-critical embedded application?
7. Types of real-time systems. (Hard, Soft, and Firm) and components of RTOS
8. Why is task prioritization critical in real-time systems?
9. How do hardware constraints affect real-time system design?
10. How does RTOS ensure predictability in task execution?
11. Which RTOS (FreeRTOS or VxWorks) is better suited for large-scale industrial applications, and why?
12. Explain the architecture of FreeRTOS and VxRTOS.
13. What factors influence task priority assignment in an embedded system?
14. How do you ensure that inter-task communication does not result in task starvation?
15. How would you handle synchronization between tasks and interrupts in a time-critical embedded application?
16. What are task dependencies, and how are they managed in RTOS? How does a scheduler handle tasks with equal priority?
17. Explain the significance of Context switching in RTOS in ES. What are the key challenges in task scheduling in embedded systems compared to general-purpose systems?
18. What are the primary goals of task scheduling in an RTOS? What are the key challenges of task synchronization in embedded systems compared to general-purpose systems?
19. Write short notes on Mutexes, semaphores, event flags, and message queues. Also, explain mutex vs Semaphores.
20. Define task priority and explain its significance in meeting deadlines. How do task synchronization and task prioritization differ from each other?
21. What are task dependencies, and how are they managed in RTOS? What strategies can handle tasks that overrun their allocated execution time?

Unit IV: Embedded System Design using VHDL (5 hrs.)
4.1 Introduction to VHDL
4.2 Different Modelling styles in VHDL for combinational and sequential circuits
4.3 Data types in VHDL
4.4 Subprogram and Packages
4.5 VHDL realization for combinational and sequential circuits

(15-22 marks) (Maybe 3 long=22 marks or 2 long=15 marks)

1. Mention different types of data type in VHDL.
2. How std_logic_vector and std_logic differ? Explain different styles of modeling in VHDL with proper examples and their differences.
3. Primary differences between Data, Behavioral, and structural modeling styles in VHDL?
4. What are the features and advantages of VHDL programming?
5. What is a test bench? Why it is necessary?

Program(Also show Wave)

1. VHDL code for Combinational circuit (Specially done in Lab)
   a. Mux or Demux
   b. Encoder or decoder
   c. Half and full adder
   d. Half and full subtractor
   e. 4. Octal to binary code converter
   f. Full adder using two half adders.
2. VHDL code for the sequential circuit (Moore and Mealy Machine)
   a. Implement a Moore state machine for an elevator control system with states for up, down, and idle.
   b. Design a sequence detector for a pattern 1101 using nonoverlapping concepts using either Moore or Melay.
   c. Any 1 flipflop
   d. Counter
3. Write a test bench code for all the examples done in the lab.

Unit V: Communication Protocols (3 hrs.)

5.1 Serial Communication:
UART, SPI, I2C

5.2 Wireless Communication:
Bluetooth, ZigBee, Wi-Fi, LoRa, GSM/GPRS

5.3 Networking:
TCP/IP Basics in Embedded Systems

(13-15 marks) (Maybe 2 long=15 marks or 1 long and 1 short notes =13 marks)

1. Design an embedded system for Wearable Health Monitoring devices.. Discuss how UART, SPI, and I²C interact with sensors, displays, and controllers.
2. How would you send an SMS using a GSM module in an embedded application? Also define LoRa, and how does it differ from other wireless communication protocols?
3. What is the role of the MOSI, MISO, SCLK, and SS lines in SPI communication?
4. Compare Wi-Fi with Zigbee and LoRa for IoT applications in terms of power consumption and range.
5. What is LoRa, and how does it differ from other wireless communication protocols? How does LoRa achieve low power consumption in embedded applications?
6. In a TCP/IP-based embedded system, how would you optimize network performance for low-power devices? How would you design a secure TCP/IP communication channel between embedded systems and remote servers?

Case Study Type Questions

1) A company is designing a smart home automation system in which various sensors (Temperature, Motion, Gas, etc.) and actuators (Light fans, door locks) communicate with a central hub. The system must support both wired and wireless communication for different devices.

Que

a) Identify and justify suitable communication protocols (Wired and wireless) for this system.

b) Compare SPI, I2C, and UART regarding data rate, complexity, and power consumption for sensor communication.

2) A healthcare company is designing a wearable device to monitor heart rate and blood oxygen levels. The device should transmit data to a smartphone application and a cloud server for long-term analysis.

Que

a) Discuss the advantages and disadvantages of using Bluetooth low energy (BLE)
b) Evaluate the impact of latency, power consumption, and data throughput or protocol selection.
c) If real-time data processing is needed, which would be more suitable and why?

3) A smart farming solution includes soil moisture sensors, weather stations, and irrigation controllers connected via an embedded system. Data should be transmitted to a central monitoring unit and accessible via mobile apps.

Que

a) Compare LoraWAN and Zigbee for long-range, low-power communication in an agricultural setting.
b) What are the trade-offs between wired and wireless communication in this scenario?

4) A wearable health monitor needs to measure and transmit real-time data. Discuss how you would integrate an SPI-based ECG sensor, an I²C-based temperature sensor, and UART communication with a smartphone.

Que

a) A medical device uses I²C for internal communication. What security measures would you implement to prevent unauthorized access or interference?

5) You are designing a system where an embedded controller communicates with multiple sensors. The sensors use I²C, but the controller needs high-speed logging via SPI. How would you integrate both protocols efficiently?
6) A smart lighting system in a building uses Zigbee to communicate between devices. How would you ensure the system can scale to support additional nodes without performance degradation?
7) A factory requires real-time equipment monitoring using an embedded TCP/IP networking system. How would you prioritize reliability and fault tolerance in this setup?
8) Design a vehicle tracking system using GPS and TCP/IP for real-time location updates. How would you ensure timely data delivery to the server?

Unit VI: Peripherals and Interfacing (4 hrs.)

6.1 Sensor Interfacing: Analog and Digital Sensors (e.g., temperature, humidity, motion)

6.2 Actuator Interfacing:

- Motor Control (DC, Stepper, Servo)
- PWM for controlling brightness, speed, etc.

6.3 Display Interfacing:

- LCD, and Seven Segment Displays

(8-13 marks) (Maybe 1 long = 8 marks or 1 long and 1 short=15 marks)

1. What is sensor interfacing, and why is it important in embedded systems? What are the primary steps involved in interfacing a sensor with a microcontroller?
2. What are the factors to consider when selecting a servo motor for a specific application, such as payload, speed, and precision? Also, mention typical steps for initializing and reading data from a sensor.
3. What are the challenges of interfacing light sensors like LDRs with ADCs in low-light conditions?
4. What are the differences between I2C and SPI protocols when interfacing with digital sensors?
5. What are the typical steps involved in calibrating a temperature sensor? What is the difference between static and dynamic calibration of sensors?
6. Why is calibration important for sensor interfacing in embedded systems? Explain Sensor interfacing and PWM
7. How do you handle communication errors (e.g., timeouts, checksum errors) when interfacing with digital sensors? How do you ensure that the data from a digital temperature or humidity sensor is calibrated and accurate over time?
8. What are the key differences between analog and digital motion sensors (e.g., PIR, ultrasonic, accelerometers), and how do they affect the interfacing process?
9. What are the primary types of motors used in embedded systems, and what are their key differences?
10. What is the difference between DC motors, stepper motors, and servo motors in terms of control and application?

Program

1. Write an AVR C program to interface a 7-segment display. Display numbers from 0 to 9 in a loop with a delay of 1 second.
2. Write an AVR C program to display your name on the first row and **"Hello!"** on the second row of a 16x2 LCD.
3. Write an AVR C program to display the message "Hello, World!" on a 16x2 LCD using 4-bit mode.
4. Write an AVR C program to display a counter that increments every second on the LCD. Use Timer1 to generate a 1-second delay.
5. Write an AVR c Program to interface a 16*2 LCD (in 4-bit mode) with an AVR microcontroller (eg. ATmega16/ATmega32). The program should

a) Initialize the LCD in 4-bit mode.
b) Display the message ‘AVR LCD TEST’.

Unit VII: Internet of Things (IoT) and Embedded Systems (3 hrs.)

7.1 Introduction to IoT Concepts and Embedded System’s Role in IoT.
7.2 IoT communication protocol: MQTT (Basic Concept)

7.3 Overview of common IoT hardware platforms: Arduino, ESP32, and Raspberry Pi. (Basic Introduction)

(13-15 marks) (Maybe 2 long=15 marks or 1 long and 1 short notes =13 marks)

1. What is IoT? How it helps in ES. Also ,Explain its applications.
2. What are the key components of IoT architecture? Briefly describe their roles.
3. Identify the key challenges faced during the implementation of the IoT system. How were these challenges addressed?
4. How was data collected, processed, and transmitted in the IoT system? Discuss the protocols or technologies used.
5. What are the security risks associated with IoT-enabled smart homes, and how can they be mitigated?
6. All the Es are IOT but all IOT are not ES justify it and why are embedded systems considered the backbone or foundation of IoT devices? What role did embedded systems play in the success of the IoT solution?
7. What factors should be considered when selecting an embedded platform (e.g., Arduino, Raspberry Pi) for IoT projects?
8. Compare Arduino, ESP32, and Raspberry Pi in terms of processing power, memory, and communication features.
9. Analyze the security and privacy challenges in IoT healthcare systems. Propose solutions using embedded technologies.
10. Design a simple smart lighting system using IoT and describe its operation.
11. How does the publish/subscribe model apply to embedded systems communication in IoT devices?
12. Write a sample code to blink an LED using Arduino.
13. Explain the built-in Wi-Fi and Bluetooth features of ESP32.
14. What is Bluetooth Low Energy (BLE), and how does ESP32 use it for communication?
15. Compare the suitability of QoS 0, QoS 1, and QoS 2 in the context of a low-power embedded sensor network.
16. What are the potential issues with using a high QoS level in resource-constrained embedded systems?
17. Explain the concept of MQTT in IOT.
18. Smart Home Automation System
    a) Describe how Arduino can be used to develop a home automation system.
    b) What sensors and actuators are required for this system?
    c)How does Arduino handle the communication between devices?
    d)What are the limitations of using Arduino for a large-scale home automation project?
19. Weather Monitoring System
    a) Explain the working of a weather monitoring system using Arduino and sensors.
    b) What communication protocols can be used to send data to the cloud?
    c) Provide sample code to interface a temperature and humidity sensor with Arduino.
20. d)How can the system be enhanced for real-time monitoring?
21. Smart Traffic Management System
    a) How can Raspberry Pi be used to design a smart traffic monitoring system?
    b) What sensors and cameras are required for this project?

22. Industrial Automation System
    a) Discuss how Raspberry Pi is used in industrial automation applications.
    b) What protocols (e.g., Modbus, MQTT) are suitable for communication in factories?
    c) How is real-time data logging achieved using Raspberry Pi?
    d) Explain how remote monitoring is enabled through Raspberry Pi's network capabilities.
23. In a smart agriculture system, sensor nodes communicate soil moisture data to a cloud server via MQTT. Data loss is acceptable during good connectivity, but reconnection should resend the most recent readings.
    a) Which QoS level suits this scenario and why?
    b) How would the embedded system manage data retransmission in case of connection loss?
24. A smart parking system using embedded devices requires each parking lot sensor to report availability status to a cloud server.
    i) How would you manage MQTT design for multiple parking lots?
    ii)How can you ensure consistent status updates during server outages?
    iii)Explain the working of a system using Arduino and sensors.
    iv)What communication protocols can be used to send data to the cloud?
    v)How can the system be enhanced for real-time monitoring?

25. In an IoT-based agricultural system, embedded devices control water distribution through actuators. Discuss:
    a) What real-time operating systems (RTOS) handle this automation?
    b) The impact of system failures and strategies for fault tolerance.