THE EMBEDDED SYSTEM DESIGN PROCESS

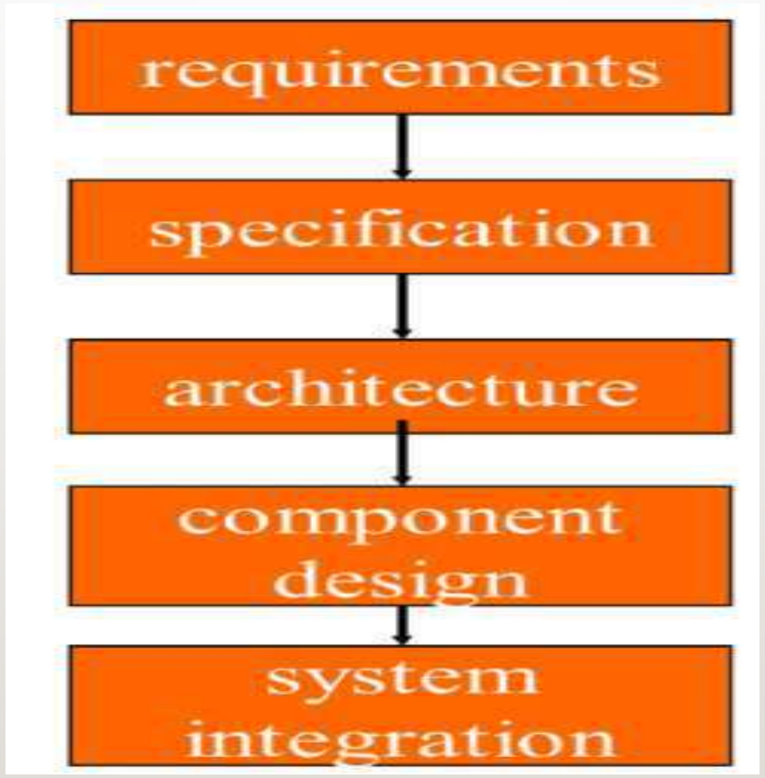
INTRODUCTION

The design process of an embedded system involves several steps, from initial requirements gathering to final implementation and testing. This presentation outlines the detailed steps involved in designing an embedded system.

DESIGN METHODOLOGIES

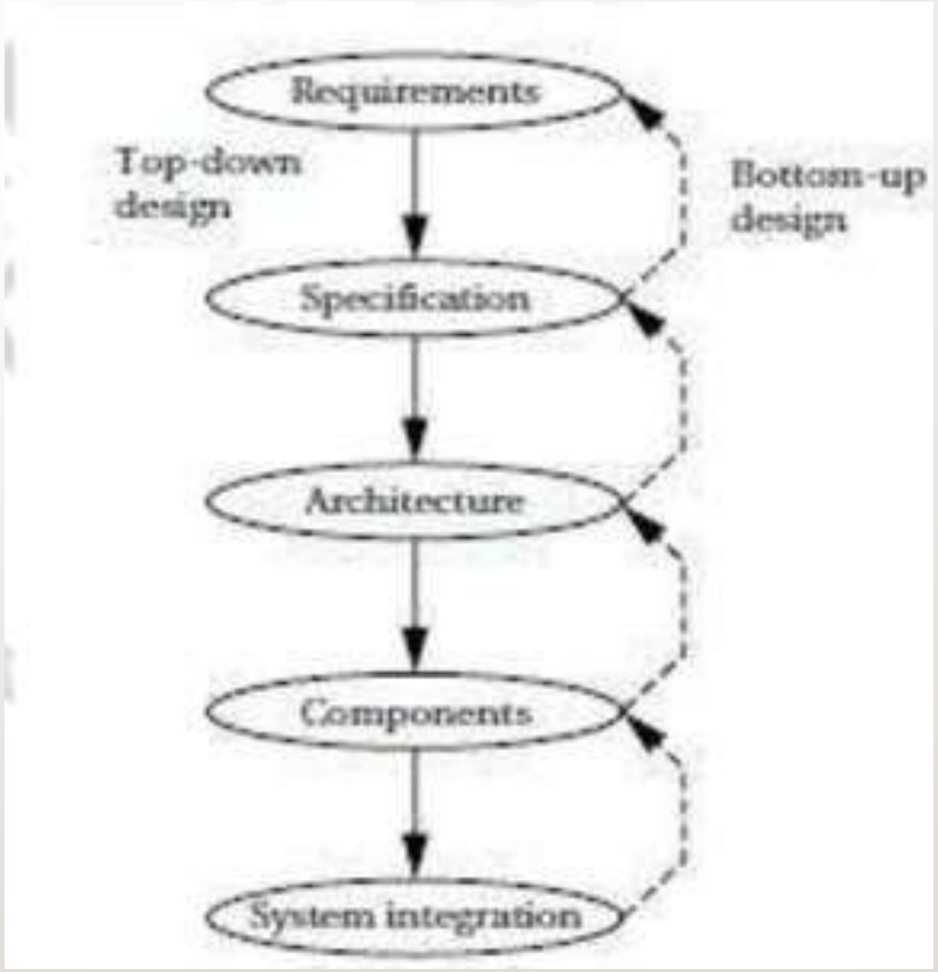
- It is a procedure for designing a system.
- A design methodology makes it much easier for members of a design team to communicate.
- It is important for optimizing performance and developing computer aided design tools.

EMBEDDED SYSTEM DESIGN PROCESS



- Top-down Design
  - It begins with the most abstract description of the system and concludes with concrete details.
- Bottom-up Design
  - It starts with components to build a system.

- In the figure the dashed line arrows indicate bottom up design. And straight line arrows indicate top-down design.



MAJOR GOALS OF DESIGN

- Manufacturing cost: The total expenses involved in producing the embedded system hardware, including components, assembly and testing.
- Performance: The ability of the system to execute tasks efficiently, accurately and within the required time constraints.
- Power consumption: The amount of electrical power the system consumes during operation.

At each step of the design process, these three tasks need to be performed.

- Analyze the design at each step to determine whether the specifications are met.
- Refine the design to add details if needed.
- Verify the design to ensure that it still meets all system goals such as cost, speed, and so on.

REQUIREMENTS – FIRST PHASE

- Before designing a system, one should know what must be designed.
- Informal descriptions that are gathered from the customer are known as requirements.
- Plain language description of user needs are called requirements.
- Requirements can be gathered in several ways like,
  - Talking directly to customers
  - Talking to marketing representatives
  - Providing prototypes to users for comment

- The requirements are refined into a specification to begin the designing of the system architecture.
- Requirements can be functional or non-functional requirements.
- Functional requirements - need output as a function of input. It defines what the system must do like tasks, behaviors, operations.
  - Examples: Sensor data collection, Real-time processing of inputs, Generating specific outputs.

- Non-functional requirements – Define constraints and quality attributes including performance, cost, physical size, weight, and power consumption.
    - Performance may be a combination of soft performance metrics such as approximate time to perform a user-level function and hard deadlines by which a particular operation must be completed. Example – Latency, throughput and real-time responsiveness
    - Cost includes the manufacturing, nonrecurring engineering and other costs of designing the system.
    - Physical size and weight are the physical aspects of the final system. These can vary greatly depending upon the application.
    - Power consumption can be specified in the requirements stage in terms of battery life.

- Sample Requirements form looks like

Name
Purpose
Inputs
Outputs
Functions
Performance
Manufacturing cost
Power
Physical size/weight

- Example for requirements form

Name	GPS moving map
Purpose	Consumer-grade moving map for driving
Inputs	power button, two control buttons
Outputs	back-lit LCD 400X600
Functions	5-receiver GPS; Three user selectable resolutions; Displays current latitude/longitude
Performance	Updates screen within 0.25 sec of movement
Manufacturing cost	$100
Power	100 MW
Physical size/weight	No more than 2” X 6” in size and 12 Ounces in weight

OUTCOMES

- A detailed Requirements Specification Document (RSD) that includes
  - Functional and non-functional requirements
  - System constraints and assumptions
  - Performance goals and benchmarks
  - High-level block diagrams for system overview

SPECIFICATION – SECOND PHASE

- The specification phase is the first and most critical step.
- It defines the system's objectives, functionality constraints and requirements.
- It serves as a blueprint for subsequent development stages.
- A well-defined specification ensures that the system meets the desired goals and expectations.
- Requirements gathered is refined into a specification.
- Specification serves as the contract between the customers and the architects.
- Specification is essential to create working systems with a minimum of designer effort.
- It must be specific, understandable and accurately reflect the customer's requirements.

Purpose of specification phase is to

  - Clearly define what the system is expected to do.
  - Identify system requirements, including functional and non-functional aspects.
  - Establish a foundation for system design, development and validation

- Key components of specification
  - System objectives: Define the purpose of the system and the problem to solve
  - Functional Requirements: Specify core functions and tasks the system must perform
  - Performance requirements: Define the desired performance metrics, such as speed, accuracy and power consumption
  - Constraints: Identify limitations such as cost, size, weight, and power consumption
  - Interface Requirements: Specify how the system will interact with external components or users
  - Environmental conditions: Define the operating environment, including temperature, humidity and vibration
  - Safety and Reliability: Outline safety features and reliable goals
  - Regulatory and Compliance Requirements: Identify standards and certifications the system must meet
  - Scalability and Future Expansion: Specify the potential for future upgrades or additional features.
  - User Requirements: Define expectations from the user’s perspective including ease of use and aesthetic appeal.

- Deliverables of the Specification Phase
  - Requirement Document: A detailed document outlining all system specifications and requirements.
  - Use Cases and Scenarios: Descriptions of how the system will be used in different situations.
  - System Architecture Outline: A high-level view of the system’s components and interactions.
  - Validation Criteria: Metrics and tests to verify the system meets specifications.

- Importance of Specification Phase

  - Acts as a contract between stakeholders, designers and developers.
  - Reduces the risk of errors or misunderstandings during later stages.
  - Ensures all stakeholders have a clear vision of the system’s objectives and limitations.

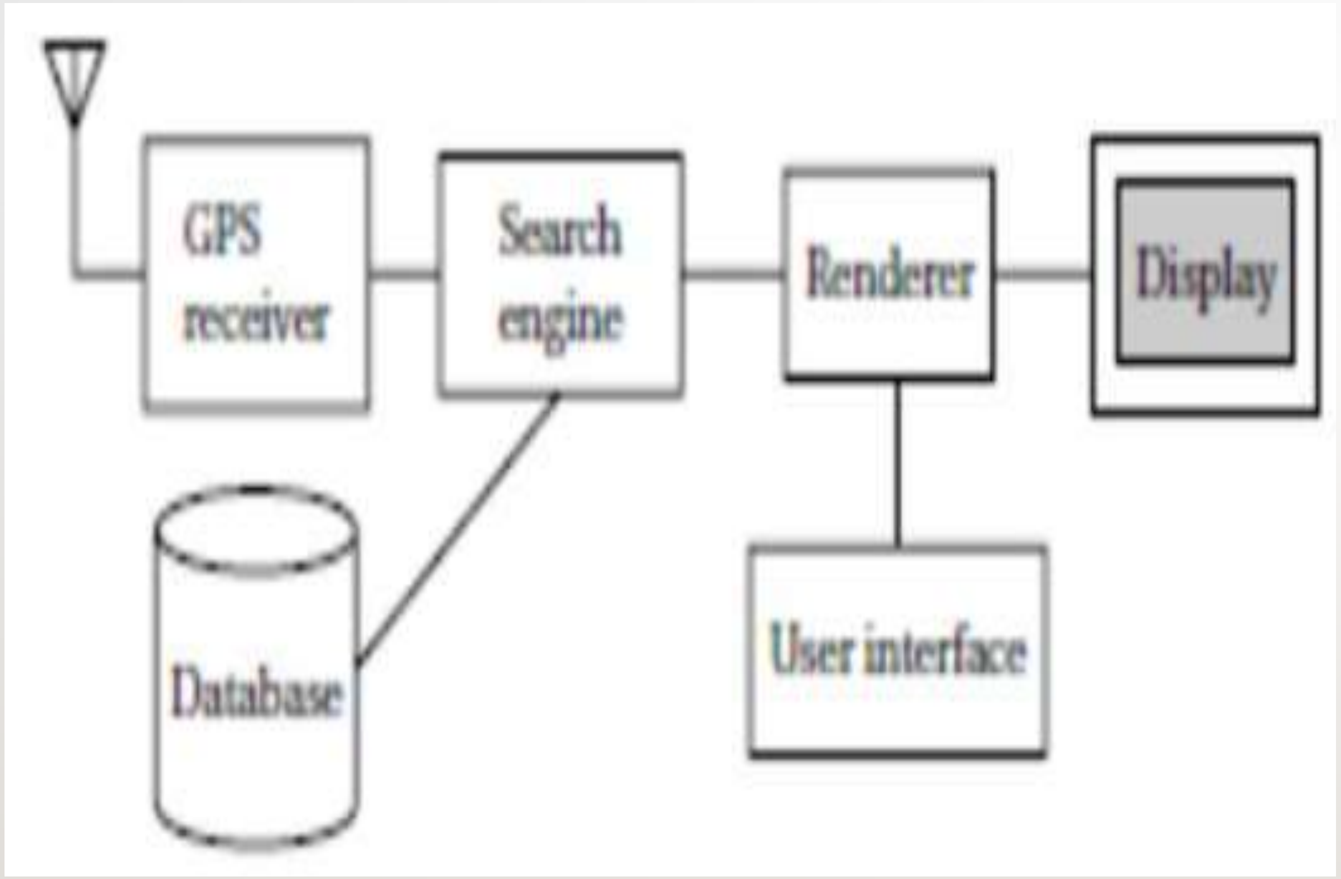
- Example:

Considering the example of the GPS system, the specification would include details for several components:

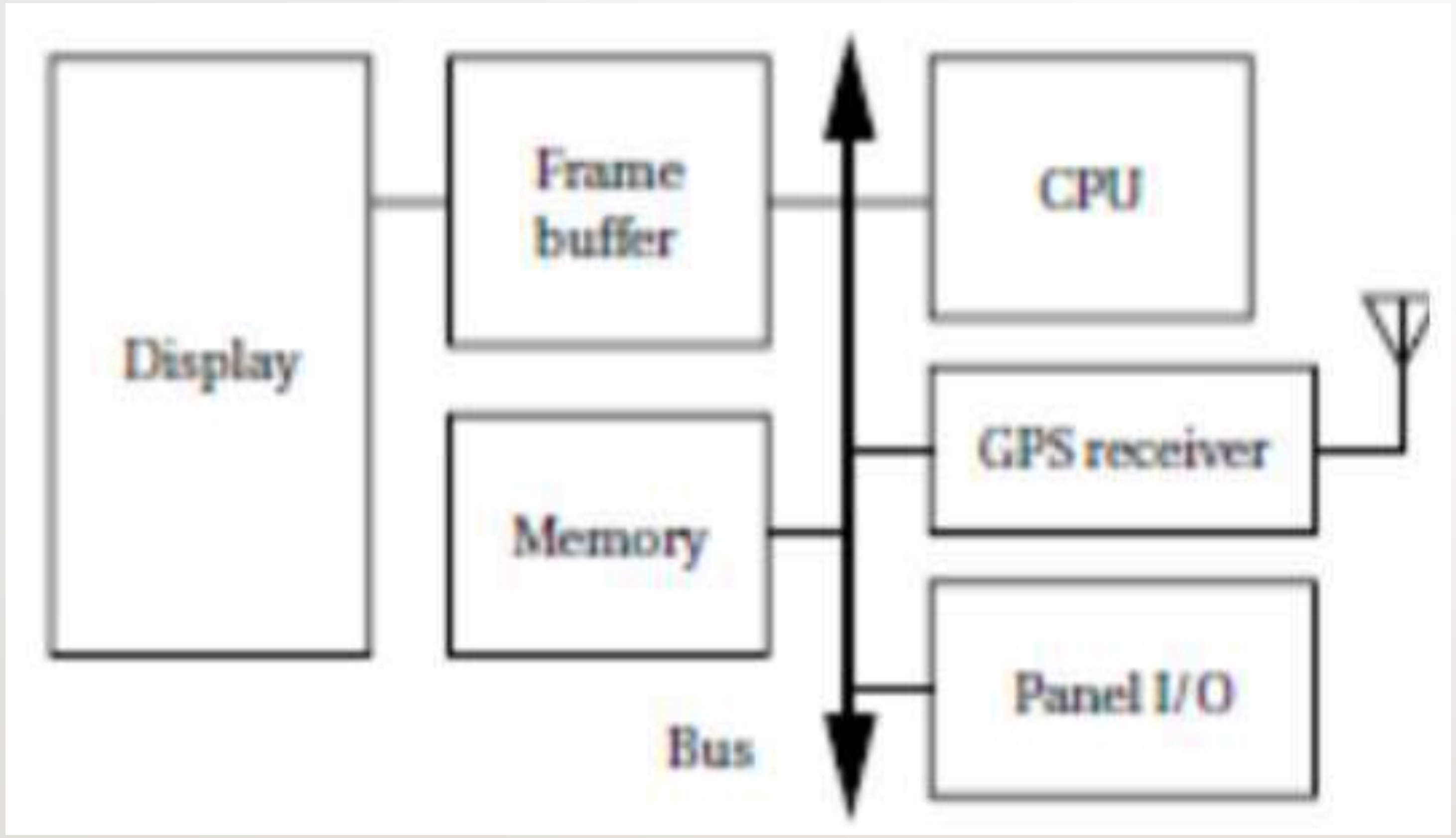
    - Data received from the GPS satellite constellation
    - Map data
    - User interface
    - Operations that must be performed to satisfy customer requests
    - Background actions

ARCHITECTURE DESIGN PHASE

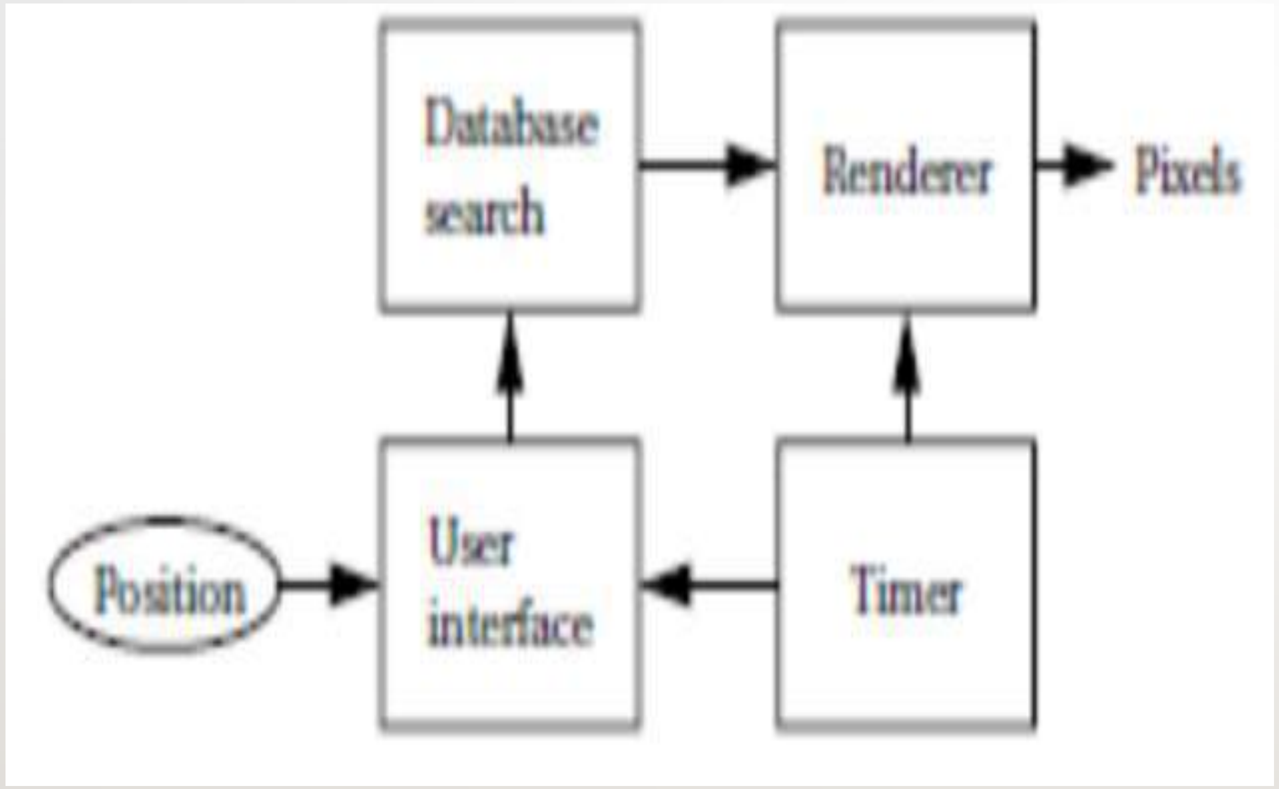
- The architecture design phase is a critical step.
- It defines system's structure including both hardware and software components.
- It provides blueprint that ensures the system meets the functional and non-functional requirements.
- Architecture is the plan for the overall structure of the system that is used later to design the components that make up the architecture.
- Architecture design gives an idea about
  - Major components need for satisfying specification
  - Hardware components and software components needed
  - Functional and non-functional requirements.



- Architecture is illustrated using block diagrams
- Example:A basic block diagram of the GPS system shows the major operations and the data flow among the blocks.
- It is an initial architecture which explains GPS navigating system.
- GPS receiver gets current position and the destination is taken from user, digital map is found in database and displayed by the renderer.
- The system block diagram may be refined into two block diagrams
  - Hardware block diagram
  - Software block diagram



- Hardware block diagram
- Hardware consists of one central CPU surrounded by memory and I/O devices.
- In this example, there are two memories one is frame buffer for the pixels to be displayed, another one is separate program/data memory for general use of CPU.
- GPS receiver is used to get the GPS coordinates
- Panel I/O is used to get the destination from the user.



- Software block diagram
- It closely follows system block diagram.
- There is a timer to control reading data from buttons and rendering data onto the screen.
- To have a truly complete architectural description, more details are required such as where units in the software block diagram will be executed, when the operations will be performed in time.

- Architectural descriptions must be designed to satisfy functional and non-functional requirements.
- Hardware and software architectures meet constraints on speed, cost and so on based on
  - Estimating the properties of the components in the block diagrams
  - Estimating accurately derives both general design and particular experience
  - Estimating all the non-functional constraints.

SYSTEM ARCHITECTURE DESIGN

1. Hardware Architecture

- Select the appropriate hardware components.
- Design the overall hardware layout.

2. Software Architecture

- Define the software structure and components.
- Determine the interaction between hardware and software.

COMPONENT DESIGN PHASE

- From the architectural description, the components that are needed can be defined.
- The component design effort builds the components that are defined in architecture and specification.
- The components includes both hardware and software modules.
- Some components are ready-made, some can be modified from existing designs and some other must be designed from scratch.

Example:

- In the moving map, GPS receiver is a predesigned hardware component.
- Topographic software is standard software module that uses standard routines to access the database.
- Printed circuit boards are the components that are to be designed.
- Custom programming is to be done.
- When creating embedded software modules, more memory space must not be allowed, system should run properly in real time.
- Power consumption is important for the hardware components.
- Memory transactions must be carefully done to avoid duplication of data.

SYSTEM INTEGRATION PHASE

- It is a critical stage in the embedded system design process.
- In this phase, all hardware and software components are combined into a single functional system.
- This phase ensures that individual modules which are developed and tested separately must work together to meet the system's overall requirements.
- Bugs are found during this phase. This phase is difficult because it uncovers problems.

Key steps in system Integration

- Component integration
  - Combine individual hardware and software components incrementally.
  - Start with essential components and progressively add modules.
- Interface testing
  - Verify that communication interfaces between hardware modules and software components function correctly.
  - Ensure compatibility between hardware and processing units.
- Subsystem integration
  - Integrate subsystems
  - Validate their combined functionality in isolation from the complete system.
- System level testing
  - Test the entire system as a unified entity
  - Validate that it meets functional, performance and reliability constraints.

- Error Detection and debugging
  - Identify and resolve integration issues, such as timing mismatches, communication failures, or hardware-software conflicts
  - Use debugging tools like oscilloscopes, logic analyzers, and software debuggers.
- System optimization
  - Optimize performance, power consumption, and resource utilization based on integration results
  - Tune system parameters for smooth operation
- Validation against requirements
  - Ensure the integrated system meets the original design specifications and application needs
  - Perform compliance testing for industry standards if applicable

Challenges in system integration

- Hardware-software integration issues
  - Timing mismatches or improper signal synchronization
  - Incompatibilities between hardware drivers and application software

- Interface conflicts
  - Errors in data exchange protocols or improper configuration of communication interfaces

- Resource constraints
  - Limited memory, processing power or I/O capabilities in embedded systems.

- Debugging complexity
  - Difficulty in pinpointing issues due to interactions between multiple components.

Tools and techniques for system integration

- Hardware debugging tools
  - Oscilloscopes, logic analyzers, and multimeters
- Software debugging tools
  - Integrated development environments (IDEs) and software emulators
- Simulation and emulation
  - Use virtual environments to simulate system behavior before hardware deployment
- Automated testing
  - Implement scripts for repetitive test cases to ensure reliability and consistency

Importance of system integration

- Ensures that the embedded system functions as intended in its real-world environment
- Identifies and resolves incompatibilities and errors early in the development process.
- Improves the reliability and performance of the final product.

System integration is a decisive step that bridges the gap between the design and deployment phases of an embedded system, ensuring all components function together harmoniously.

Summary

In the embedded system design process,

- First information is collected and refined in the requirement step.
- Specification phase uses the refined information to describe the functions of the system that reflects the customer requirements and also serves as the contract between customer and designer.
- Architecture design implements the functions described by the specification phase. It describes the components that are needed including both hardware and software.
- Component design phase builds all needed components described by the architectural design phase.
- System integration phase integrates all the components that are built including software and hardware.