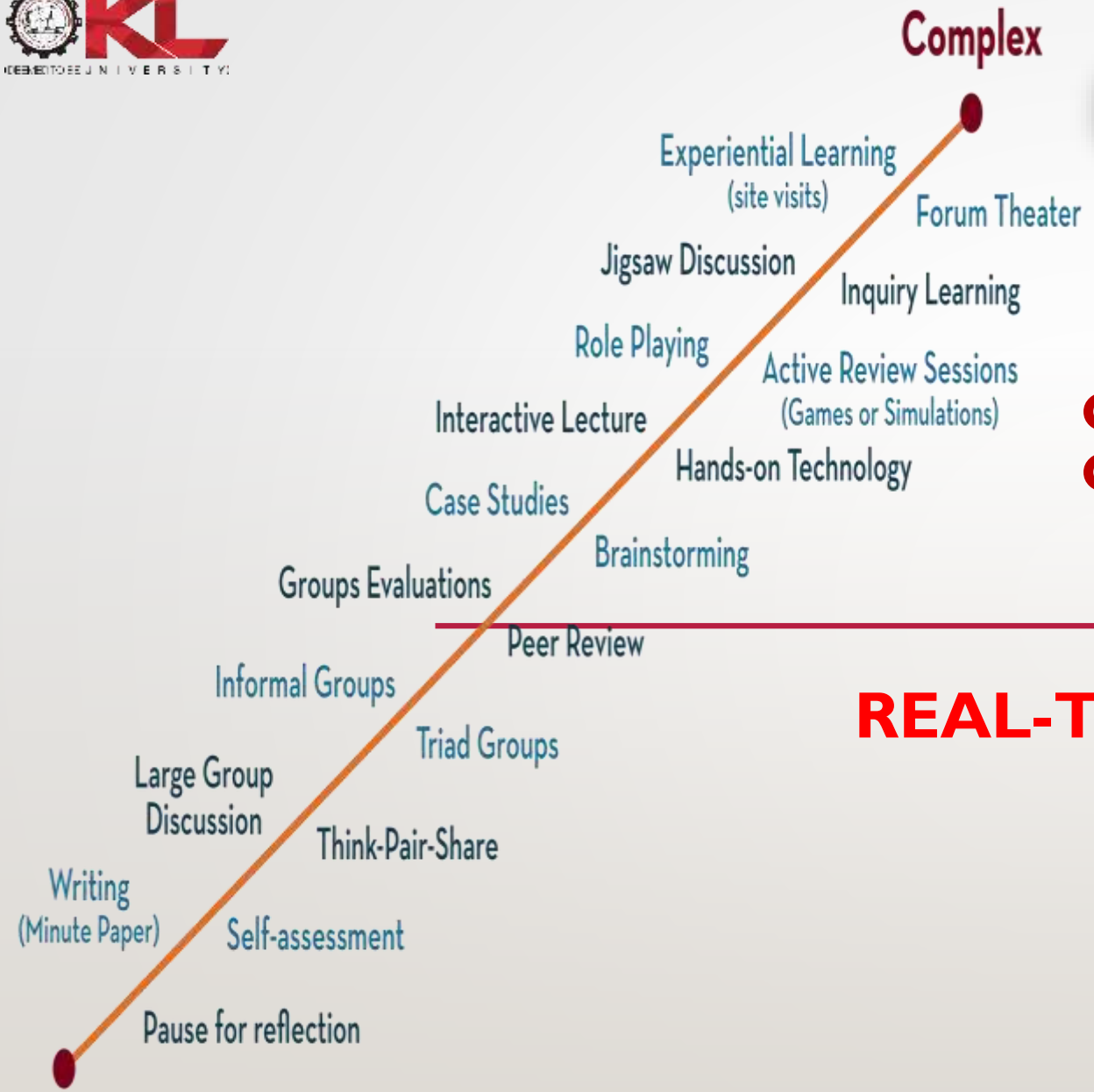


CO4

Session - 6

COURSE NAME : OPERATING SYSTEMS

COURSE CODE : 23CS2104R/A

REAL-TIME AND EMBEDDED SYSTEMS, FAULT TOLERANCE.

AIM OF THE SESSION

To familiarize students with the basic concept of real-time and Embedded Systems.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Demonstrate what is meant by Real-time Systems.
2. Demonstrate what is meant by Embedded Systems.
3. Describe the types of Real-time Systems and Embedded Systems.
4. Describe the Advantages and Disadvantages of Real-time and Embedded Systems.

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Defines what Is Realtime Systems.
2. Defines what Is Embedded Systems.
3. Summarize the Role of Realtime and Embedded Systems.

REAL-TIME SYSTEMS

A real-time system means that the system is subjected to real-time, i.e., the response should be guaranteed within a specified timing constraint or the system should meet the specified deadline. A real-time system responds to events or stimuli within a specific time frame, ensuring predictable behavior.

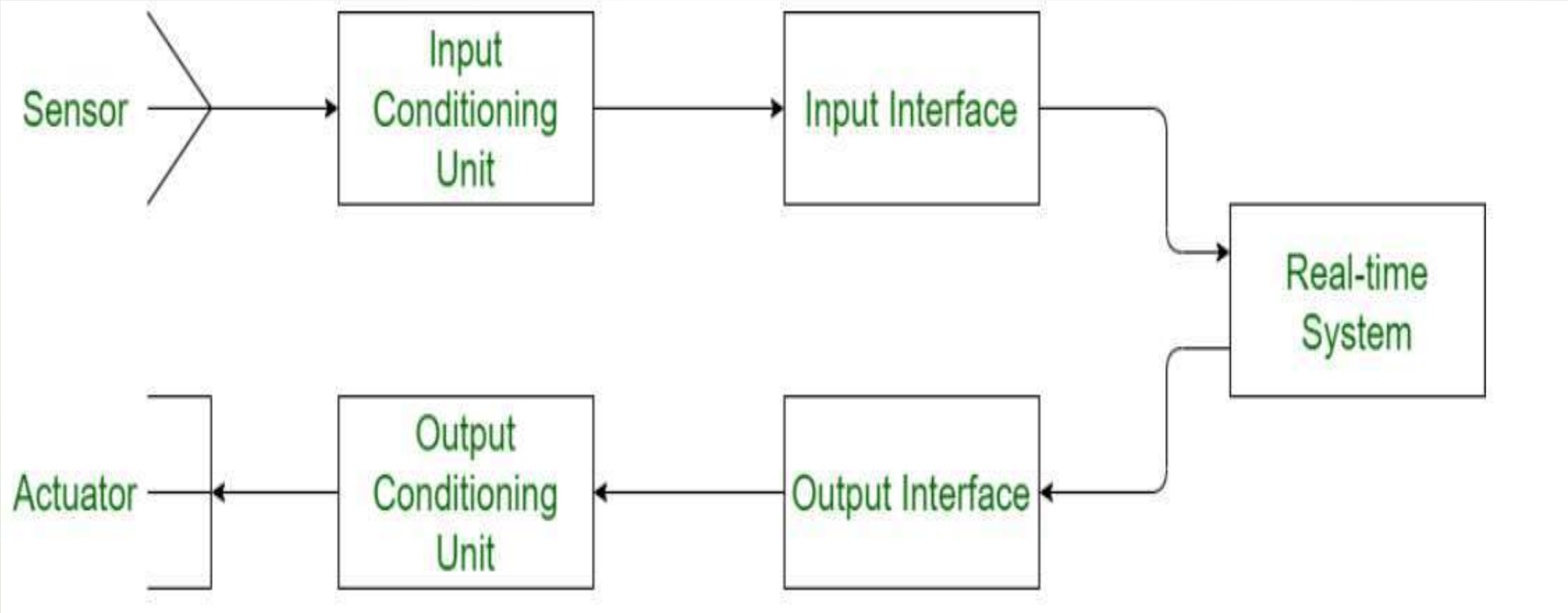
Characteristics:

- **Deterministic:** The system must perform actions within defined time limits.
- **Time-bound:** It must respect deadlines.
- **Predictability:** The system's behavior is predictable under all operating conditions.

Examples:

- Aircraft control systems (flight management).
- Robotics (precise motor control).

STRUCTURE OF REAL-TIME SYSTEMS



TYPES OF REAL-TIME SYSTEMS BASED ON TIMING CONSTRAINTS

1. **Hard real-time system:** This type of system can never miss its deadline. Missing the deadline may have disastrous consequences. The usefulness of results produced by a hard real-time system decreases abruptly and may become negative if tardiness increases (How late a real-time system completes its task concerning its deadline). **Example:** Flight controller system.
2. **Soft real-time system:** This type of system can miss its deadline occasionally with some acceptably low probability. Missing the deadline has no disastrous consequences. The usefulness of results produced by a soft real-time system decreases gradually with an increase in tardiness. Example: Telephone switches.
3. **Firm Real-Time Systems:** These are systems that lie between hard and soft real-time systems. In firm real-time systems, missing a deadline is tolerable, but the usefulness of the output decreases with time. Examples of firm real-time systems include online trading systems, online auction systems, and reservation systems.

REAL-TIME SYSTEM REQUIREMENTS

Timing Constraints

- Each task must be completed within a specific time (deadline).
- Example: In healthcare, a pacemaker must send electrical impulses at precise intervals.

Reliability and Availability

- The system should always be operational (high uptime), as failure could have severe consequences.

Predictability

- The system should behave consistently under all circumstances to ensure proper operation.

COMPONENTS OF A REAL-TIME SYSTEM

Real-Time Operating System (RTOS):

- Manages hardware and software resources in real-time, ensuring tasks meet deadlines.

Task Scheduling:

- Determines when and how tasks are executed, ensuring the system meets its time requirements.

Resource Management:

- Allocates processor time, memory, and other resources efficiently among tasks.

REAL-TIME SCHEDULING ALGORITHMS

Rate Monotonic Scheduling (RMS):

- Tasks with shorter execution times are given higher priority. Ideal for static task sets.

Earliest Deadline First (EDF):

- Tasks with the nearest deadlines are executed first, providing flexibility in task scheduling.

Priority Inversion:

- A situation where lower-priority tasks hold resources needed by higher-priority tasks, causing delays.

REAL-TIME SYSTEM DESIGN CHALLENGES

Predicting Task Execution Time:

- Real-time systems must guarantee that each task is completed within its deadline. Accurately predicting execution time is crucial.

Handling External Interrupts:

- Interrupts from hardware or sensors must be processed immediately, as they affect real-time system behavior.

Deadlocks and Priority Inversions:

- Special attention is needed to avoid deadlocks and prevent priority inversion from delaying high-priority tasks.

APPLICATIONS OF REAL-TIME SYSTEMS

Automotive Systems:

- Real-time systems control engine operations, transmission, and safety features like ABS.

Healthcare Devices:

- Pacemakers and diagnostic systems must respond to medical events in real time.

Defense Systems:

- Missile guidance, radar, and defense mechanisms rely on real-time computing to process inputs and outputs quickly.

TERMS RELATED TO REAL-TIME SYSTEM

- **Job:** A job is a small piece of work that can be assigned to a processor and may or may not require resources.
- **Task:** A set of related jobs that jointly provide some system functionality.
- **Release time of a job:** It is the time at which the job becomes ready for execution.
- **Execution time of a job:** It is the time taken by the job to finish its execution.
- **Deadline of a job:** It is the time by which a job should finish its execution. Deadline is of two types: absolute deadline and relative deadline.
- **Response time of a job:** It is the length of time from the release time of a job to the instant when it finishes.

TERMS RELATED TO REAL-TIME SYSTEM

- The maximum allowable response time of a job is called its **relative deadline**.
- The **absolute deadline** of a job is equal to its relative deadline plus its release time.
- Processors are also known as active resources. They are essential for the execution of a job. A job must have one or more processors in order to execute and proceed towards completion. Example: computer, transmission links.
- Resources are also known as passive resources. A job may or may not require a resource during its execution. Example: memory, mutex.
- Two resources are identical if they can be used interchangeably else they are heterogeneous.

ADVANTAGES

- Real-time systems provide immediate and accurate responses to external events, making them suitable for critical applications such as air traffic control, medical equipment, and industrial automation.
- They can automate complex tasks that would otherwise be impossible to perform manually, thus improving productivity and efficiency.
- Real-time systems can reduce human error by automating tasks that require precision, accuracy, and consistency.
- They can help to reduce costs by minimizing the need for human intervention and reducing the risk of errors.
- Real-time systems can be customized to meet specific requirements, making them ideal for a wide range of applications.

DISADVANTAGES

- Real-time systems can be complex and difficult to design, implement, and test, requiring specialized skills and expertise.
- They can be expensive to develop, as they require specialized hardware and software components.
- Real-time systems are typically less flexible than other types of computer systems, as they must adhere to strict timing requirements and cannot be easily modified or adapted to changing circumstances.
- They can be vulnerable to failures and malfunctions, which can have serious consequences in critical applications.
- Real-time systems require careful planning and management, as they must be continually monitored and maintained to ensure they operate correctly.

WHAT IS AN EMBEDDED SYSTEM

An embedded system is a dedicated computer system designed to perform a specific function or a set of functions.

Characteristics:

- - Task-specific
- - Operates in real-time
- - Resource-constrained

COMPONENTS OF AN EMBEDDED SYSTEM

- Hardware Components:
  - Microcontroller / Microprocessor
  - Memory (ROM, RAM, EEPROM)
  - Sensors
  - Actuators
- Software Components:
  - Firmware
  - Real-Time Operating System (RTOS)
  - Device Drivers

TYPES OF EMBEDDED SYSTEMS

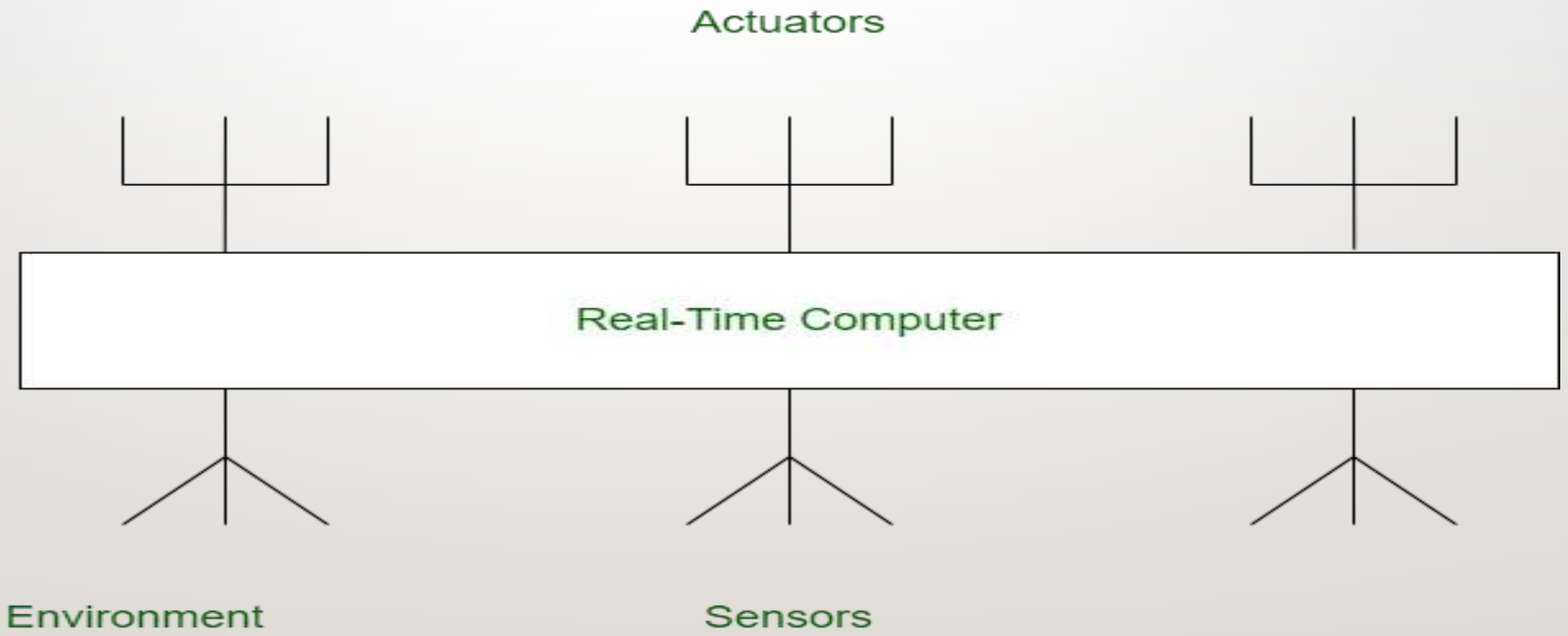
- Small Scale: 8-bit or 16-bit microcontrollers
  - Example: Digital Watches
- Medium Scale: 16-bit or 32-bit processors
  - Example: Home Appliances
- Large Scale: High-end processors like ARM
  - Example: Smartphones

REAL-TIME EMBEDDED SYSTEMS

Definition: Systems that meet precise timing requirements.

- Hard Real-Time: Missing deadlines results in failure
  - Example: Pacemakers
- Soft Real-Time: Deadlines are flexible
  - Example: Multimedia Streaming

STRUCTURE OF EMBEDDED REAL-TIME SYSTEM



MICROCONTROLLER VS. MICROPROCESSOR

- Microcontroller: Single-chip integrating CPU, memory, and peripherals.
  - Example: Arduino
- Microprocessor: General-purpose CPU requiring external peripherals.
  - Example: Intel x86

EMBEDDED SYSTEM SOFTWARE

- Embedded Firmware: Software tightly coupled with hardware.
- Languages: C, C++, Assembly
- RTOS: Manages real-time tasks and synchronization.

COMMUNICATION INTERFACES

- Common Protocols:
  - I2C: Two-wire communication
  - SPI: High-speed communication
  - CAN: Automotive systems

APPLICATIONS OF EMBEDDED SYSTEMS

- Consumer Electronics: Smartphones
- Automotive Systems: ABS, Airbags
- Healthcare: Medical Devices
- Industrial Automation: Robotics

CHALLENGES IN EMBEDDED SYSTEMS DESIGN

- Challenges:
  - Resource Constraints
  - Real-Time Performance
  - Power Consumption

FUTURE TRENDS IN EMBEDDED SYSTEMS

- Trends:
  - IoT: Internet of Things
  - Edge Computing: Data processing close to the source
  - AI in Embedded Systems.

ACTUATOR

- The actuator is the device that is the reverse of the sensor. The actuator is used to convert electrical events into physical signals while the sensor is used to do the reverse job. It may convert electrical signals into physical events or characteristics according to the requirements of the user.
- It takes input from the system and gives output to the environment.
- The output obtained from the actuator may be in the form of any physical action.
- Some of the commonly used actuators are heaters and motors.

SENSOR

- Sensor is used to sense the environment from time to time.
- It is used to convert physical events or characteristics into electrical signals.
- This is a hardware device that takes input from the environment and gives output to the system.
- The sensed data from the environment is processed to determine the corrective actions necessary.

REAL-TIME OPERATING SYSTEM

- Real-time **operating systems (RTOS)** are used in environments where a large number of events, mostly external to the computer system, must be accepted and processed in a short time or within certain deadlines.
- such applications are industrial control, telephone switching equipment, flight control, and real-time simulations.
- With an RTOS, the processing time is measured in tenths of seconds. This system is time-bound and has a fixed deadline.

EMBEDDED OPERATING SYSTEM

- An embedded operating system is a specialized operating system (OS) designed to perform a specific task for a device that is not a computer.
- The main job of an embedded OS is to run the code that allows the device to do its job.
- The embedded OS also makes the device's hardware accessible to software that is running on top of the OS.

FAULT TOLERANCE

- **Fault tolerance** is a process that enables an operating system to respond to a failure in hardware or software.
- This fault-tolerance definition refers to the system's ability to continue operating despite failures or malfunctions.
- Fault tolerance can be built into a system to remove the risk of it having a single point of failure.
- The key benefit of fault tolerance is to minimize or avoid the risk of systems becoming unavailable due to a component error.

HARDWARE FAULT-TOLERANCE TECHNIQUES

- Making a hardware fault-tolerance is simple as compared to software.
- Fault-tolerance techniques make the hardware work properly and give correct results even if some fault occurs in the hardware part of the system.
- Build in Self Test(BST) and Triple Modular Redundancy(TMR) two techniques used for hardware fault-tolerance

HARDWARE FAULT-TOLERANCE TECHNIQUES

- **BIST**: When the system detects a fault, it switches out the faulty component and switches in the redundancy of it. The system reconfigures itself in case of fault occurrence.
- **TMR:** Three redundant copies of critical components are generated and all these three copies are run concurrently.

SOFTWARE FAULT-TOLERANCE TECHNIQUES

- Software fault-tolerance techniques are used to make the software reliable in the condition of fault occurrence and failure.
- Techniques used in Software Fault-tolerance Techniques
  - N-version Programming
  - Recovery Blocks
  - Check-pointing and Rollback Recovery

SOFTWARE FAULT-TOLERANCE TECHNIQUES

- **N-version Programming:** In N-version programming, all the redundant copies are run concurrently and the result obtained is different from each processing.
- **Recovery Blocks:** In the recovery block, all the redundant copies are not run concurrently and these copies are run one by one.
- **Check-pointing and Rollback Recovery:** In this technique, the system is tested each time when we perform some computation. This technique is useful when there is processor failure or data corruption.

THANK YOU

Team – Operating System