

Introduction to Embedded Systems

Introduction to Embedded Systems

(A concise, easy-to-read study guide)

1. What Is an Embedded System?

Aspect	Description
Definition	A computer system designed to perform a dedicated function (or a set of functions) within a larger mechanical or electrical device.
Key Characteristics	• Real-time operation • Limited resources (CPU, memory, power) • Tight integration with hardware • Often invisible to the end-user • Microwave oven controller • Automotive engine control unit (ECU)
Examples	• Wearable fitness tracker • Industrial PLC (Programmable Logic Controller) • Smart thermostat

2. Core Components

Component	Role	Typical Choices
Processor / MCU	Executes the software; may be a microcontroller, microprocessor, DSP, or FPGA.	ARM Cortex-M, AVR, PIC, TI MSP430, RISC-V, Intel x86 (rare)
Memory	Stores program code and runtime data.	• Flash/ROM – program storage • RAM – volatile data • EEPROM/FRAM – non-volatile parameters
Peripherals	Interface to the external world.	GPIO, UART, SPI, I ² C, ADC/DAC, PWM, CAN, USB, Ethernet, Wi-Fi, Bluetooth

Introduction to Embedded Systems

Component	Role	Typical Choices
Power Supply	Provides stable voltage; often constrained by battery life.	Linear regulators, switching DC?DC converters, energy?harvesting modules
Sensors / Actuators	Convert physical phenomena to electrical signals (sensors) and vice?versa (actuators).	Temperature sensor, accelerometer, motor driver, relay, LED, speaker
Software Stack	Firmware, real?time operating system (RTOS), drivers, middleware, application code.	Bare?metal, FreeRTOS, Zephyr, μ C/OS, Linux (for higher?end devices)

3. Classification of Embedded Systems

Category	Typical Performance	Example
Small?scale	8?bit MCU, a few KB RAM/Flash, low power	Digital watches, simple toys
Medium?scale	32?bit MCU, 100?KB–1?MB RAM, moderate peripherals	Home appliances, automotive sensors
Complex (System?on?Chip)	32/64?bit CPU, >1?MB RAM, rich I/O, OS support	Smartphones, automotive infotainment, industrial robots

4. Design Flow Overview

1. Requirement Analysis

- Define functional specifications, real?time constraints, power budget, cost targets.

2. System Architecture

- Choose processor family, memory map, peripheral set, and communication interfaces.

3. Hardware Design

- Schematic capture, PCB layout, component selection, power?management design.

Introduction to Embedded Systems

4. Software Development

- Bootloader ? Low?level drivers ? RTOS / kernel ? Middleware ? Application.

5. Integration & Testing

- Unit tests, hardware?in?the?loop (HIL), functional safety analysis (ISO 26262, IEC 61508).

6. Production & Maintenance

- Firmware update strategy (OTA, JTAG), reliability monitoring, field diagnostics.

5. Real?Time Requirements

Concept	Meaning	Typical Metric
---------	---------	----------------

**Hard real?