Embedded system

An embedded system is a combination of computer hardware and software designed for a specific function. Embedded systems might also function within a larger system. These systems can be programmable or have a fixed functionality. Embedded systems are used today to control numerous devices. For example, they're used in industrial machines, consumer electronics, agricultural and processing industry devices, automobiles, medical devices, cameras, digital watches, household appliances, airplanes, vending machines, toys and mobile devices.

Embedded systems typically contain a microprocessor -- or a microcontroller-based system, memory and input/output (I/O) devices, all of which share a dedicated function within a larger system. While embedded systems are computing systems, they can range from having no user interface (UI) -- for example, on devices designed to perform a single task -- to complex graphical user interfaces (GUIs), such as in mobile devices. UIs can include buttons, light-emitting diodes (LEDs) and touchscreen sensing. Some systems use remote user interfaces as well.

According to Global Markets Insight, the embedded systems market was valued at $110.3 billion in 2023 and is predicted to grow to more than $190 billion by 2032. Chip manufacturers for embedded systems include many well-known technology companies, such as Apple, IBM, Intel and Texas Instruments. The expected growth is partially due to the continued investment in artificial intelligence (AI), mobile computing and the need for chips designed for high-level processing.

Examples of embedded systems

Embedded systems are used in a wide range of technologies across an array of industries. Some examples include the following:

- **Automobiles.** Modern cars commonly consist of many computers, or embedded systems, designed to perform different tasks within the vehicle. Some of these systems perform basic utility functions and others provide entertainment or user-facing functions. As modern cars become more computerized, the number of embedded systems increases. Some embedded systems in consumer vehicles include cruise control, backup sensors, suspension control, navigation systems, alarm systems and airbag systems.
- **Mobile phones.** These consist of many embedded systems, including GUI software and hardware, operating systems (OSes), cameras, microphones, and Universal Serial Bus I/O modules.
- **Industrial machines.** These contain embedded systems, such as sensors, and can be embedded systems themselves. Industrial machines often have embedded automation systems that perform specific monitoring and control functions.

- **Medical equipment.** These contain embedded systems such as sensors and control mechanisms. Medical equipment, such as industrial machines, must also be user-friendly so that human health isn't jeopardized by preventable machine mistakes. This means these systems often include a more complex OS and GUI designed for an appropriate UI.
- **Fitness trackers.** These wearable devices contain embedded systems that collect data on the user such as heart rate, blood and oxygen levels and number of steps.

How does an embedded system work?

Embedded systems always function as part of a complete device. They're low-cost, low-power consuming, small computers that are embedded in other mechanical or electrical systems. Generally, they comprise a processor, power supply, and memory and communication ports. Embedded systems use the communication ports to transmit data between the processor and peripheral devices -- often, other embedded systems -- using a communication protocol. The processor interprets this data with the help of minimal software stored in the memory. The software is usually highly specific to the function that the embedded system serves.

Embedded systems often communicate with outside systems in a device, enabling data exchange and control functions.

The processor might be a microprocessor or microcontroller. Microcontrollers are simply microprocessors with peripheral interfaces and integrated memory included. Microprocessors use separate integrated circuits for memory and peripherals instead of including them on the chip. Both can be used, but microprocessors typically require more support circuitry than microcontrollers because they're less integrated into the microprocessor. The term *system-on-a-chip i*s often used. SoCs typically include multiple processors and interfaces on one chip. They're often used for high-volume embedded systems. Some examples of SoC types are the application-specific integrated circuit (ASIC) and the field-programmable gate array (FPGA).

Often, embedded systems are used in real-time operating environments and use a real-time operating system to communicate with the hardware. Near-real-time approaches are suitable at higher levels of chip capability, defined by designers who have increasingly decided the systems are generally fast enough and the tasks tolerant of slight variations in reaction. In these instances, stripped-down versions of the Linux OS are commonly deployed, although other OSes have been pared down to run on embedded systems, including Embedded Java and Microsoft Windows IoT -- formerly Microsoft Windows Embedded.

Embedded system designers often also use compilers, assemblers and debuggers to develop embedded system software.

Characteristics of embedded systems

The main characteristic of embedded systems is that they're task-specific. They often include the following additional characteristics:

- They typically consist of hardware, software and firmware.
- They can be embedded in a larger system to perform a specific function, as they're built for specialized tasks within the system, not various tasks.
- They can be either microprocessor-based or microcontroller-based -- both are integrated circuits that give the system compute power.
- They often use ASIC and FPGA SoCs.
- They're often used for sensing and real-time computing in internet of things (IoT) devices, which are devices that are internet-connected and don't require a user to operate.
- They can vary in complexity and function, which affects the type of software, firmware and hardware they use.
- They're often required to perform their function under a time constraint to keep the larger system functioning properly.

Structure of embedded systems

Embedded systems vary in complexity but, generally, consist of the following three main elements:

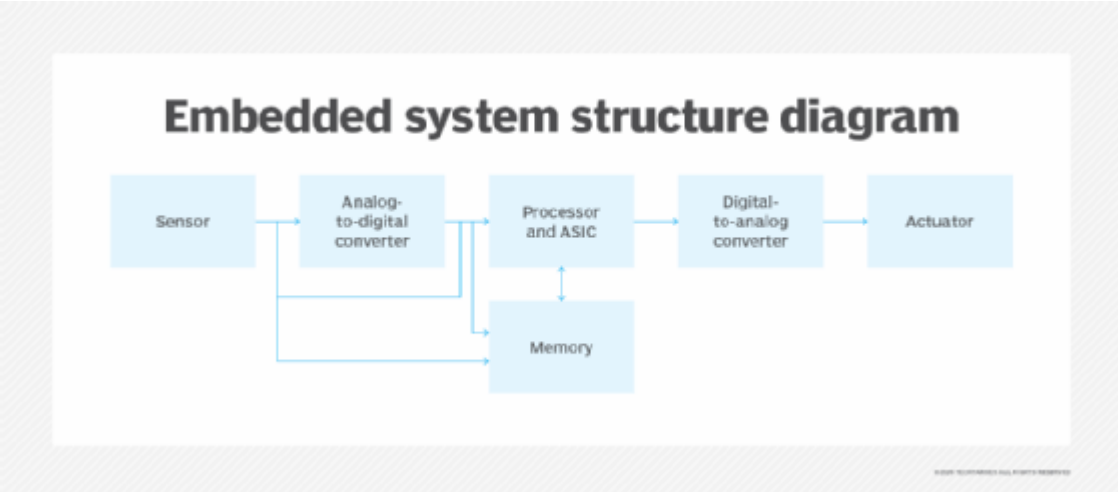
- **Hardware.** The hardware of embedded systems is based around microprocessors and microcontrollers. Microprocessors are similar to microcontrollers and, typically, refer to a central processing unit (CPU) that's integrated with other basic computing components, such as memory chips and digital signal processors. Microcontrollers have those components built into one chip.
- **Software and firmware.** Software for embedded computing systems can vary in complexity. However, industrial-grade microcontrollers and embedded IoT systems usually run simple software that requires little memory.
- **RTOSes.** These aren't always included in embedded systems, especially smaller-scale systems. RTOSes define how the system works by supervising the software and setting rules during program execution.

In terms of hardware, a basic embedded system consists of the following elements:

- **Sensors.** These components convert physical sense data into an electrical signal.
- **Analog-to-digital converters.** A-D converters change an analog electrical signal into a digital one.
- **Processors.** These process digital signals and store them in memory.

- **Digital-to-analog converters.** D-A converters change the digital data from the processor into analog data.
- **Actuators.** These components control the mechanical motion of the embedded system by converting electrical signals into physical actions.

The sensor reads external inputs, the converters make that input readable to the processor, and the processor turns that information into useful output for the embedded system.



The structure of an embedded system shows the flow of data from a sensor through an analog-to-digital converter, a processor, memory, a digital-to-analog converter and an actuator.

Types of embedded systems

Embedded system types differ in their functional requirements. They include the following:

- **Mobile embedded systems** are small systems that are designed to be portable. Digital cameras, smartphones and laptops are examples.
- **Networked embedded systems** are connected to a network to provide output to other systems. Examples include home security systems and point-of-sale systems.
- **Standalone embedded systems** aren't reliant on a host system. Like any embedded system, they perform a specialized task. However, they don't necessarily belong to a host system, unlike other embedded systems. A calculator or MP3 player are examples.
- **Real-time embedded systems** give the required output in a defined time interval. They're often used in medical, industrial and military sectors because they're responsible for time-critical tasks. A traffic control system is an example.

Embedded systems can also be categorized by the following performance requirements:

- **Small-scale embedded systems** often use no more than an 8-bit microcontroller.

- **Medium-scale embedded systems** use a larger 16-32-bit microcontroller and often link microcontrollers together.
- **Sophisticated-scale embedded systems** often use several algorithms that result in software and hardware complexities and might require more complex software, a configurable processor and a programmable logic array.

There are several common embedded system software architectures, which become necessary as embedded systems grow and become more complex in scale. These include the following:

- **Simple control loops** call subroutines, which manage a specific part of the hardware or embedded programming.
- **Interrupt controlled systems** have two loops: a main one and a secondary one. Interruptions in the loops trigger tasks.
- **Cooperative multitasking** is essentially a simple control loop located in an application programming interface.
- **Preemptive multitasking or multithreading** is often used with an RTOS and features synchronization and task-switching strategies.

Very large-scale integration (VLSI) describes the complexity of an integrated circuit (IC). VLSI is the process of embedding hundreds of thousands of transistors into a chip, whereas large-scale integration (LSI) microchips contain thousands of transistors, medium-scale integration (MSI) contains hundreds of transistors, and small-scale integration (SSI) contains tens of transistors. Ultra-large-scale integration (ULSI) refers to placing millions of transistors on a chip.

VLSI circuits are a common feature of embedded systems. Many ICs in embedded systems are VLSIs, and the use of the VLSI acronym has largely fallen out of favor.

Debugging embedded systems

Embedded systems differ from the OSes and development environments of other larger-scale computers in how they handle debugging. Usually, developers working with desktop environments can run both the code being worked on and separate debugger applications that can monitor the embedded system that programmers generally can't.

An embedded system's circuit board can feature multiple electronic components and wiring, which can include a processor, power supply, memory and communication ports.

Some programming languages run on microcontrollers with enough efficiency that rudimentary interactive debugging is available directly on the chip. In addition, processors

often have CPU debuggers that can be controlled and, thus, control program execution via the JTAG industry standard or similar debugging port.

In many instances, however, programmers need tools that attach a separate debugging system to the target system via a serial or other port. In this scenario, the programmer can see the source code on the screen of a general-purpose computer, just as they would in the debugging of software on a desktop computer.

A separate, frequently used approach is to run software on a PC that emulates the physical chip in software. This essentially makes it possible to debug the performance of the software as if it were running on an actual physical chip.

A simple way to debug embedded applications is to use a general-purpose I/O pin. This verifies that a specific line of code in an application is being executed.

Another basic debugging tool is a source-level debugger, which enables users to walk through their code, pause and check program memory or variables.

Logic analyzers are another common and useful debugging tool. They can read waveforms from multiple signals at a time, while also being able to decode that data from various standard interfaces.

Broadly speaking, embedded systems have received more attention to testing and debugging because numerous devices using embedded controls are designed for situations where safety and reliability are top priorities.

History of embedded systems

Embedded systems date back to the 1960s. Charles Stark Draper developed an integrated circuit in 1961 to reduce the size and weight of the Apollo Guidance Computer, the digital system installed on the Apollo Command Module and Lunar Module. The first computer to use integrated circuits, it helped astronauts collect real-time flight data.

In 1965, Autonetics, now a part of Boeing, developed the D-17B, the computer used in the Minuteman I missile guidance system. It's widely recognized as the first mass-produced embedded system. When the Minuteman II went into production in 1966, the D-17B was replaced with the NS-17 missile guidance system, known for its concentrated use of integrated circuits. In 1968, the first embedded system for a vehicle was released; the Volkswagen 1600 used a microprocessor to control its electronic fuel injection system.

By the late 1960s and early 1970s, the price of integrated circuits dropped and usage surged. The first microcontroller was developed by Texas Instruments in 1971. The TMS1000 series,

which became commercially available in 1974, contained a 4-bit processor, read-only memory and random-access memory, or RAM, and it initially cost around $2 each in bulk orders.

Also, in 1971, Intel released what's widely recognized as the first commercially available processor, the 4004. The 4-bit microprocessor was designed for use in calculators and small electronics, though it required external memory and support chips. The 8-bit Intel 8008, released in 1972, had 16 KB of memory; the Intel 8080 followed in 1974 with 64 KB of memory. The 8080's successor, the x86 series, was released in 1978 and is still largely in use today.

In 1987, the first embedded OS, the real-time VxWorks, was released by Wind River, followed by Microsoft's Windows Embedded CE in 1996. By the late 1990s, the first embedded Linux products began to appear. Today, Linux is used in almost all embedded devices.

Throughout the 1990s and 2000s, processing power increased due to the transition from 8- and 16-bit microcontrollers to 32- and 64-bit processors.

The 2010s saw an increased focus on security features in embedded devices, possibly driven by the rise of IoT and connected devices.

Today, due to technological advancements, embedded systems have also begun to integrate with AI and machine learning (ML) systems. Also called embedded AI, this is the integration of AI into resource-limited devices such as smartphones or autonomous vehicles.

Embedded system trends

While some embedded systems can be relatively simple, others are becoming more complex and can either supplant human decision-making or offer capabilities beyond what a human could provide. For instance, some aviation systems, including those used in drones, can integrate sensor data and act upon that information faster than a human could, permitting new kinds of operating features.

The embedded system is expected to continue growing rapidly, driven in large part by IoT. Expanding IoT applications, such as wearables, drones, smart homes, smart buildings, video surveillance, three-dimensional printers and smart transportation, are expected to fuel embedded system growth.

Other embedded system trends include the following:

- **AI and ML.** This is a currently growing trend of integrating AI and ML systems into devices such as smartphones, autonomous vehicles, industrial automation devices and wearable devices.

- **Edge computing.** Edge computing, which pushes the processing of data closer to the source device, is also becoming more prevalent in embedded systems, as it can lower latency and bandwidth usage -- especially in real-time applications.
- **Security.** As security becomes an increasing concern for many, security features such as encryption and secure boot mechanisms are being integrated into embedded systems.
- **Increased connectivity.** Continued improvements in Bluetooth and 5G technologies provide higher bandwidth and lower latency for embedded systems.
- **Quantum computing.** Integrating quantum computing with embedded systems could offer better security through quantum cryptography, improved optimization and advanced problem-solving. Practical applications, however, are still emerging.