

Embedded System Architecture - CSEN 701

Module 1: *Introduction to Embedded Systems*

Lecture 01: *Definition, Characteristics, and Applications*

Dr. Eng. Catherine M. Elias

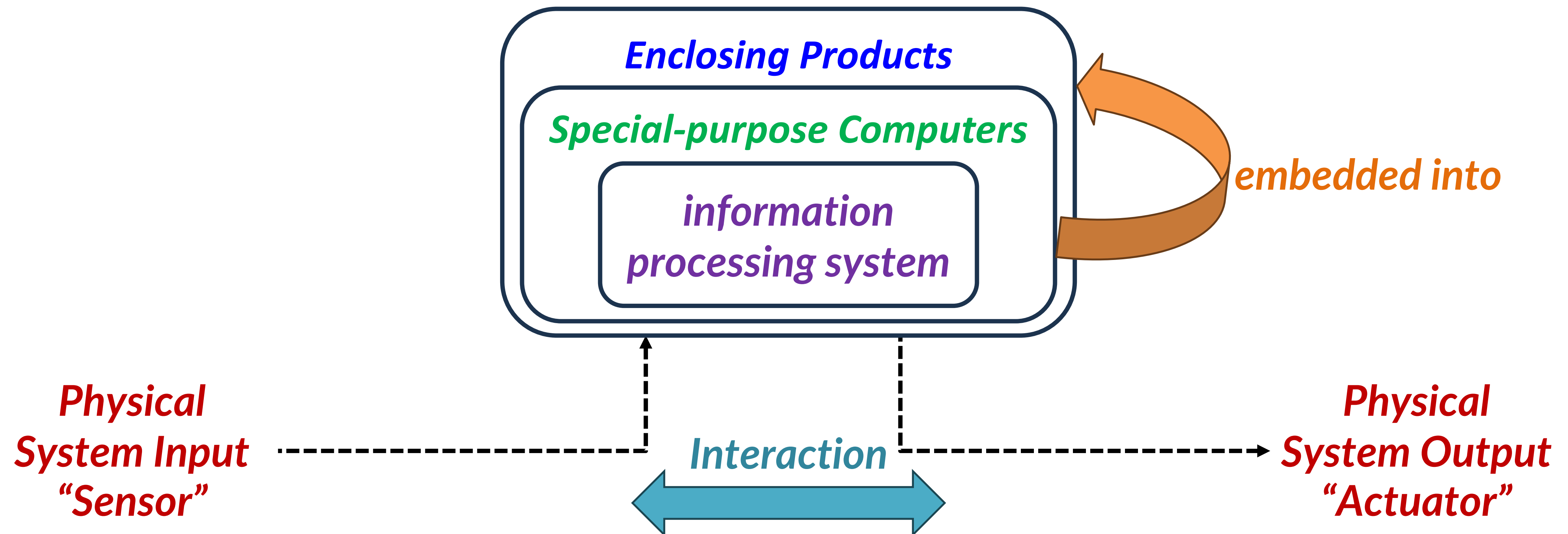
catherine.elias@guc.edu.eg

*Lecturer, Computer Science and Engineering,
Faculty of Media Engineering and Technology, German University in Cairo*

- Embedded Systems
- Real-time Systems
- The big Picture
- References

What does it mean?

Formal Definition:



“ES is special-purpose computers embedded into enclosing products, whose job is not primarily information processing, but rather the interaction with physical processes.”

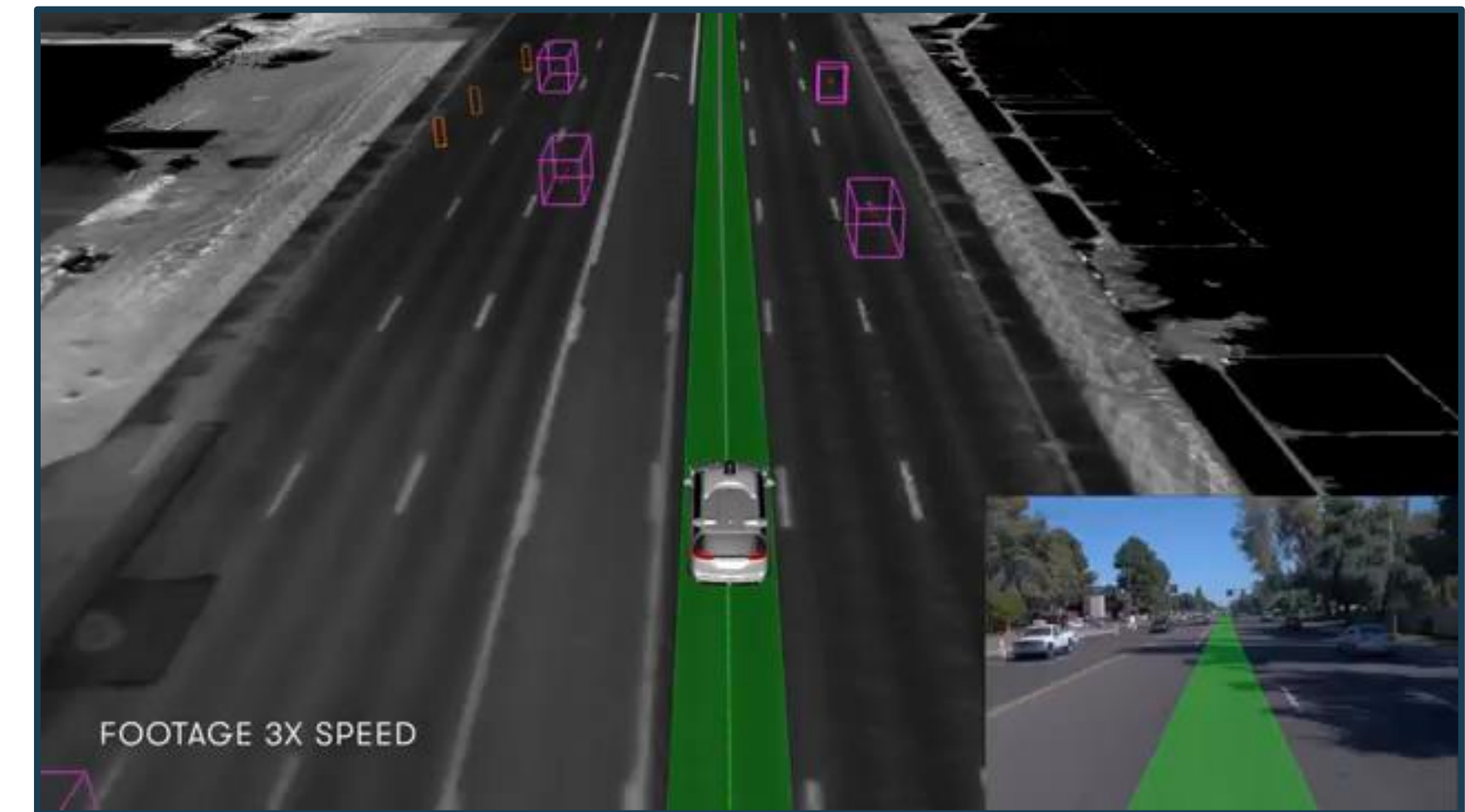
What does it mean?

- ES is a microcomputer system embedded in a larger system and designed for one or two services. [1]
- ES is a microprocessor-based system that is built to control and perform a dedicated function or range of functions and is not designed to be programmed by the end user in the same way that a PC is [2].
- An embedded system is a combination of computer hardware, software and perhaps additional parts, either mechanical or electronic—designed [3].

Medical Industry



Automotive Industry



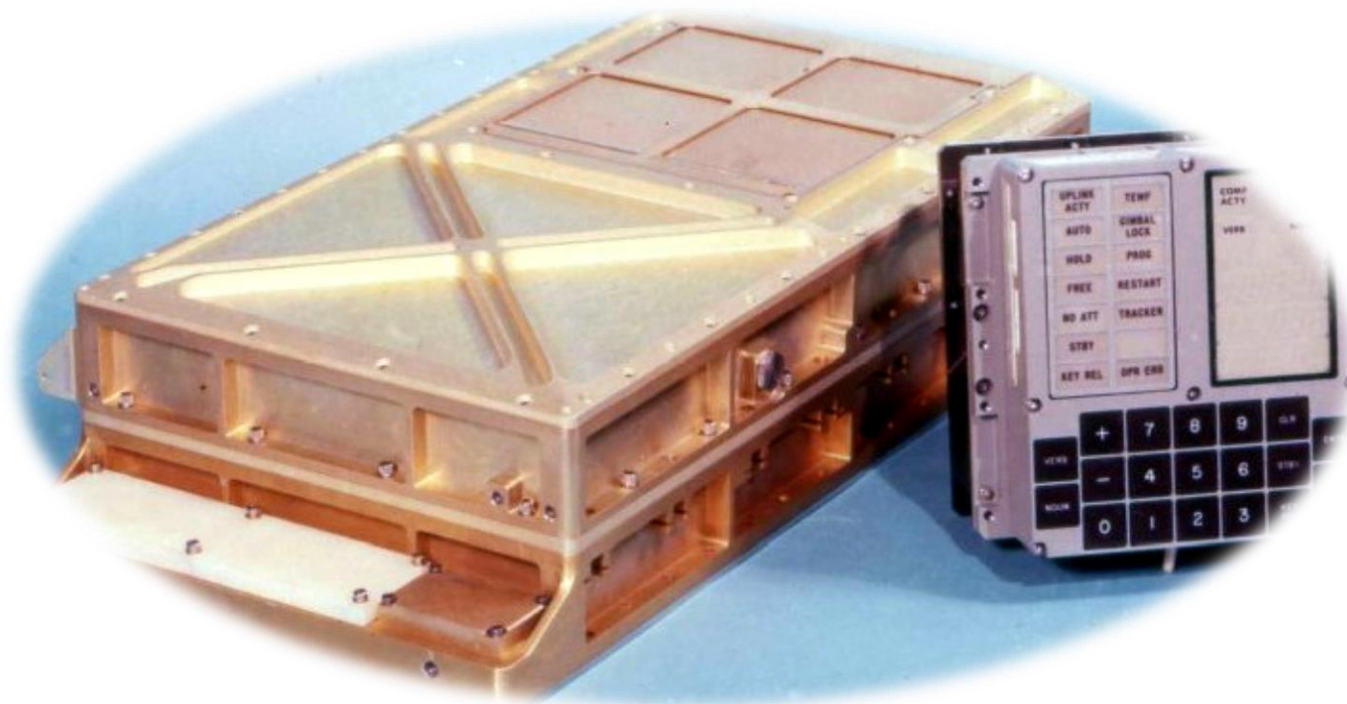
Goods Transportation Industry



ES-based Applications



Over the Years



1960's

1970's

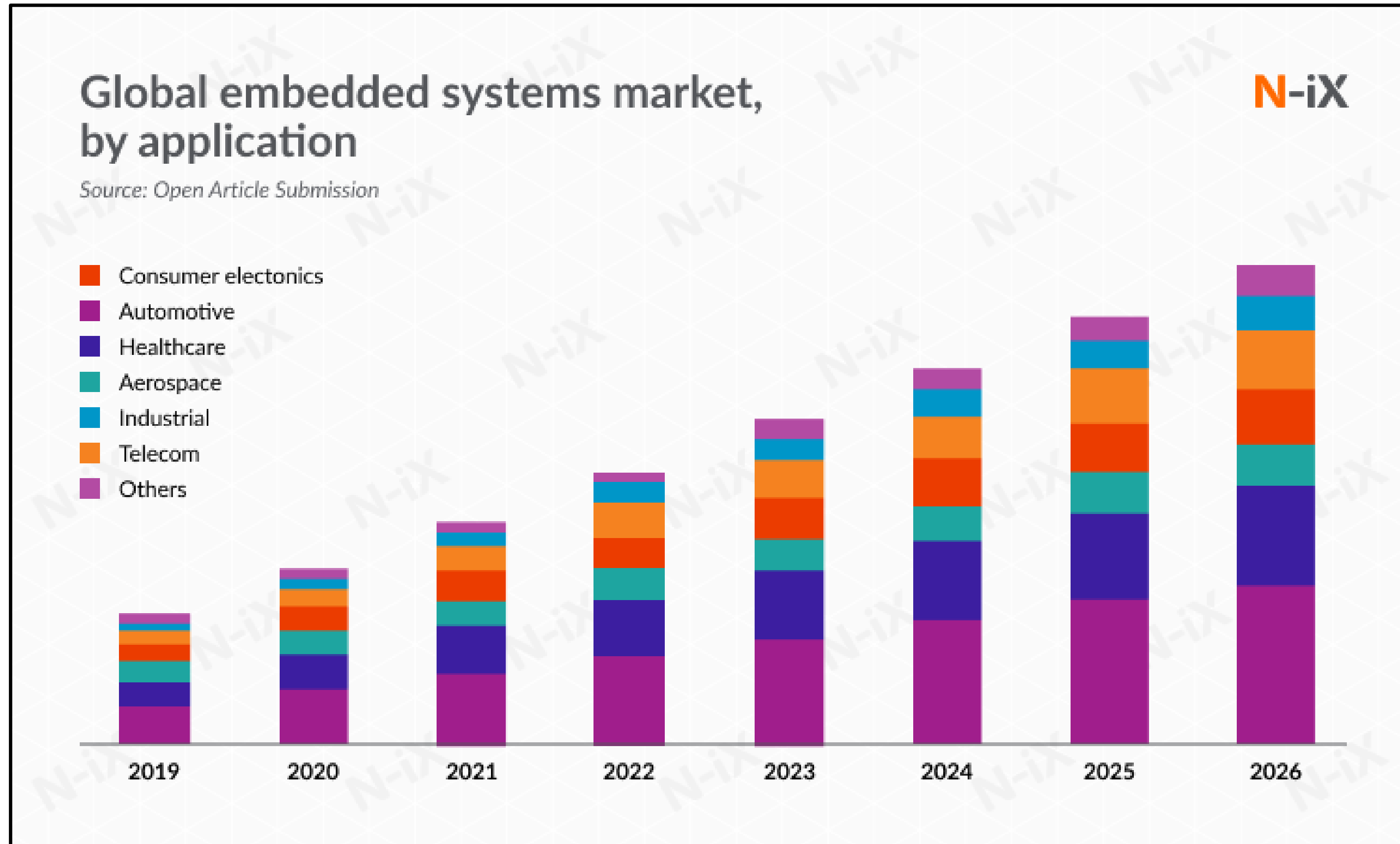
1980's

1990's

2000's



Over the Years: Industry Market



Characteristics

- Dynamical: **interacts with physical processes** through sensors and actuators.
- Heterogeneous: include both **hardware** and **software** components.
- Networked: **concurrent** and **distributed**.
- Computational: executes both **arithmetical** and **non-arithmetical** operations
- Reactive: responds at the speed of the environment (**timing matters**).
- **Safety and Reliability**: Ensuring system safety and reliability, especially in critical applications, is a priority.

Challenges

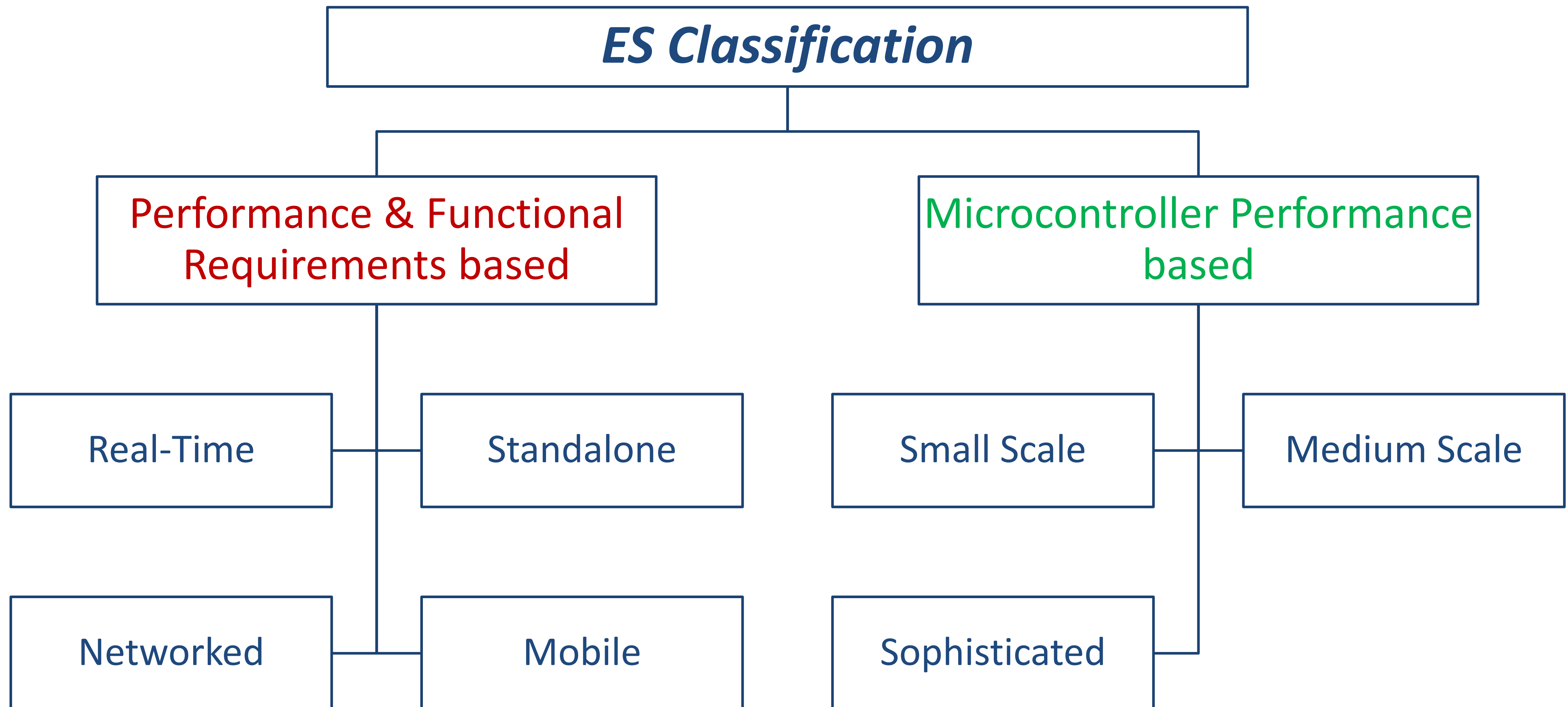
- **Power Efficiency:** Optimizing power consumption is crucial for extending battery life and minimizing energy usage, especially in battery-operated devices.
- **Real-Time Operation:** Many embedded systems must meet strict real-time requirements, responding to external events within specified timeframes.
- **Complexity Management:** Managing increasing complexity while meeting resource constraints can be challenging as embedded systems evolve.
- **Security:** Protecting embedded systems from unauthorized access and cyber threats is essential, particularly as they become more interconnected.
- **Long Product Lifecycles:** Ensuring ongoing support, software updates, and component availability for extended periods is vital for systems with long lifecycles.
- **Resource Optimization:** Efficiently utilizing limited processing power, memory, and storage resources while achieving desired functionality.

Constraints

- **Limited Resources:** restricted processing power, memory, storage, and energy, requiring efficient resource management.
- **Real-Time Requirements:** operate in real-time, which demands deterministic responses and low-latency performance.
- **Cost Limitations:** Meeting budget constraints is crucial, especially in consumer electronics, requiring a balance between cost and features.
- **Size and Form Factor:** Physical constraints, such as size, weight, and form factor, dictate the design and integration of components.
- **Environmental Challenges:** Operating in harsh environments with extreme temperatures, humidity, and vibrations may necessitate robust design considerations.
- **Security and Compliance:** Compliance with regulatory standards and the implementation of robust security measures can be challenging due to resource limitations.
- **Interoperability:** Ensuring compatibility and seamless integration with other hardware and software components in larger systems can be complex, particularly in heterogeneous environments

General purpose computer Vs. Embedded system

	General purpose computer	Embedded system
Purpose	Multi -purpose	Single function
Flexibility	Highly flexible and can be adapted for different purposes through software installations and configuration changes.	Rigid in terms of functionality. They are pre-programmed and are not easily adaptable for other purposes without significant hardware or software modifications.
Hardware Complexity	have more powerful and complex hardware components, including general-purpose CPUs, larger memory capacities, and various I/O options.	minimal hardware required for their specific tasks. They often use specialized processors and have limited memory and I/O options.
Operating Systems	run full-fledged operating systems like Windows, macOS, Linux, IOS, Android	run specialized, lightweight operating systems , or they may even run without an operating system, using firmware directly.
Constraint	Low or no resource constraint	High constraints in Size, power, cost, memory, realtime
Performance	Faster and better	Fixed runtime requirement
User Interface	External Interface Can have keyboard, display, mouse, touch screen	Minimal or no interface Integrated into the real world with buttons, sensors, Leads, LCDs, Bluetooth system



ES Classification

- **Real-time Embedded Systems**

- Perform a task in real-time
- Offer quick responses under critical situations.
- Examples: Flight Controller System, traffic light control.

- **Standalone Embedded Systems**

- Process digital/analog input signals into digital output
- Less complex
- Examples: digital watches and video games consoles.

- **Networked Embedded Systems**

- Operate through a network interface
- Communication to network happens through LAN, WAN, or other protocols
- May be wired or wireless
- Examples: ATM machines, weather monitoring systems

- **Mobile Embedded Systems**

- Small, portable and easy-to-carry
- Work on restricted memory space
- Constantly evolving to get into a miniature model
- Examples: mobile phones, digital camera

ES Classification

- **Small Scale System**

- Entry-level embedded system with no design complexity involves
- Work with 8-bit (8051) or 16-bit (80196) microcontrollers
- Most of these systems are battery operated
- Hardware and software complexity is very low due to the small size of microcontrollers
- Requires small memory as it deals with a small amount of data
- Ex: robotic arm controller, electronic toys, automatic coffee vending machines, thermometer.

ES Classification

- **Medium Scale System**

- System is designed using 16-bit or 32-bit microcontrollers
- Offers better speed than small scale Embedded system
- Hardware and software complexity is present
- Run through Microcontrollers and digital signal processors
- Requires comparatively more memory power than small scale ones
- Along with microcontrollers, you need application-specific operating systems
- Ex: Routers, ATMs, music systems, pagers

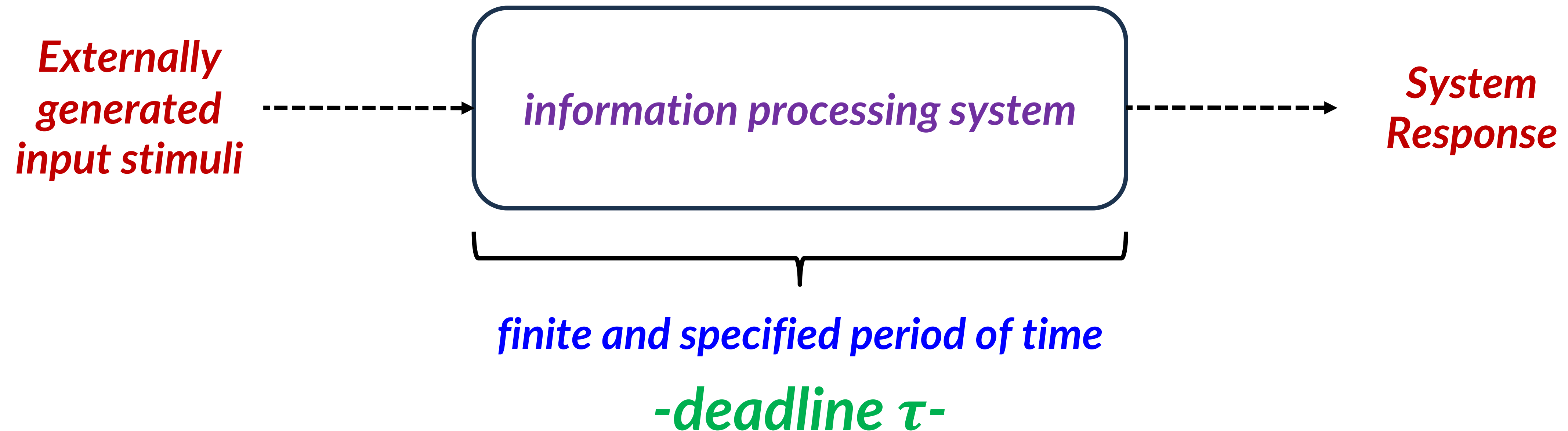
ES Classification

- ***Sophisticated*** Systems

- Make use of 32-bit or 64-bit microcontroller and multi-core processors
- Requires very high memory power
- Power consumption is the highest among rest types of embedded systems
- Hardware and software complexities are enormous
- Speed is a major concern
- Some applications, like satellite systems, also involve real-time operating system (RTOS)
- Ex: mobile systems, washing machines, digital watches, LAN cards, multimedia systems

What does it mean?

Formal Definition:



A real time system is any information processing system which has to respond to externally generated input stimuli within finite and specified period of time (deadline)

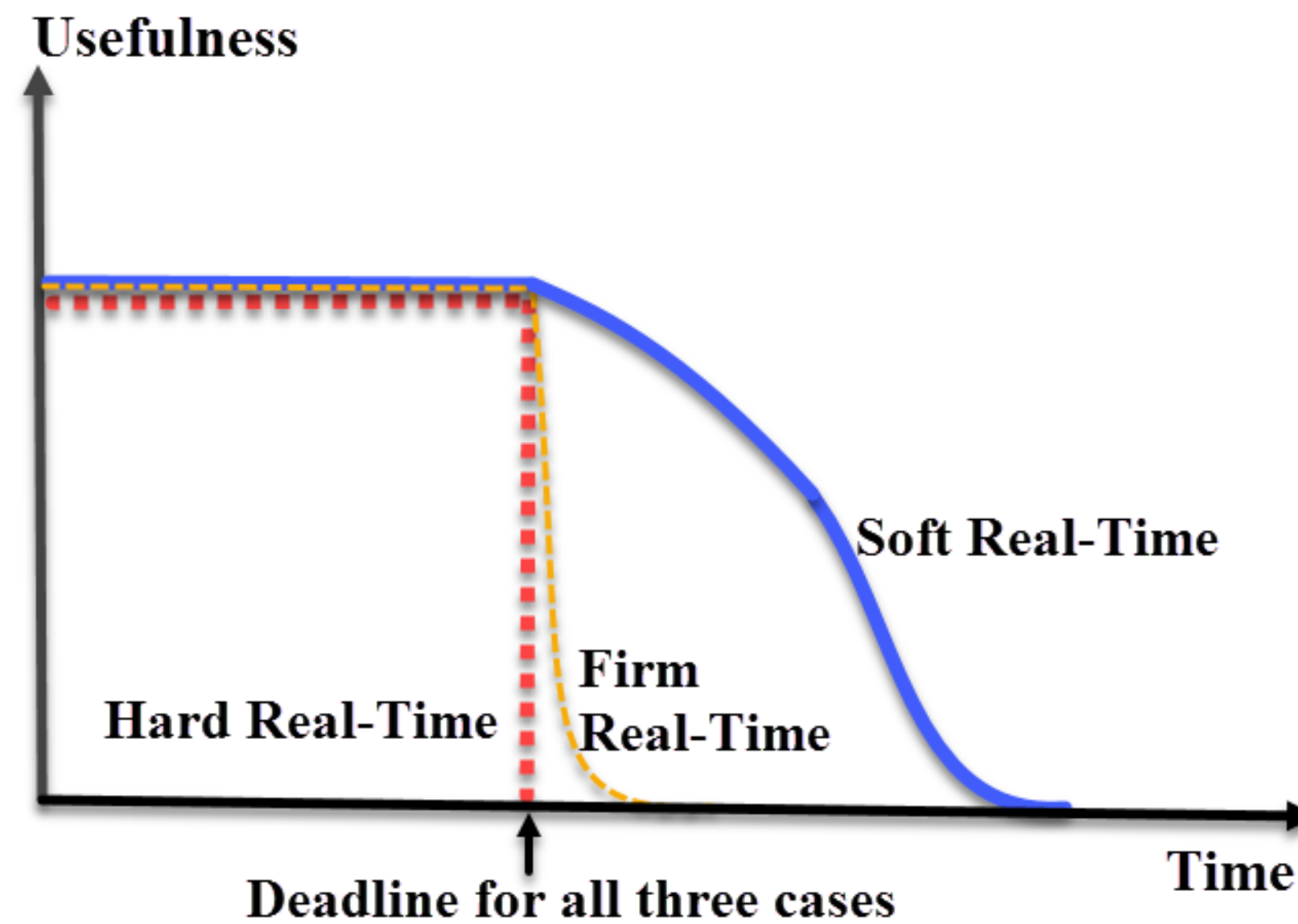
RT Systems Types

Soft
RT Systems

Firm
RT Systems

Hard
RT Systems

Weakly Hard
RT Systems



Types: **Soft** RT Systems

- This type of system **can miss its deadline occasionally** with some acceptably low probability.
- Missing the deadline have **no disastrous consequences**.
- The **usefulness** of results produced by a soft RT system **decreases** gradually with an **increase in tardiness**.
- There is some **cost** associated to overrunning, often connected to **Quality-of-Service** (QoS)



***Tardiness means how late a real-time system completes its task with respect to its deadline.*

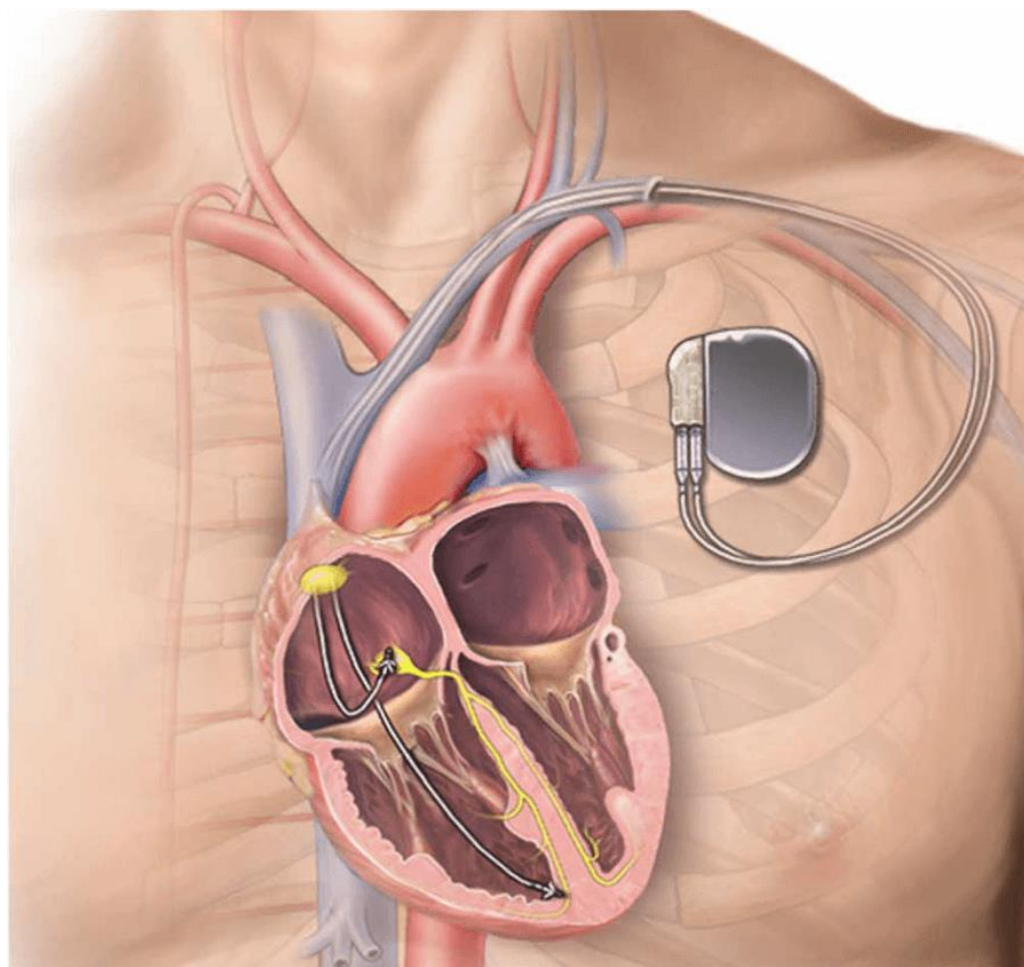
Types: **Firm** RT Systems

- These are systems that lie **between** hard and soft real-time systems.
- In firm real-time systems, **missing a deadline** is **tolerable**,
- The computation is **obsolete** if the job is **not finished on time**.
- The **usefulness** of the output **decreases** with **time**.
- **Cost** may be interpreted as **loss of energy/time/money**.



Types: **Hard** RT Systems

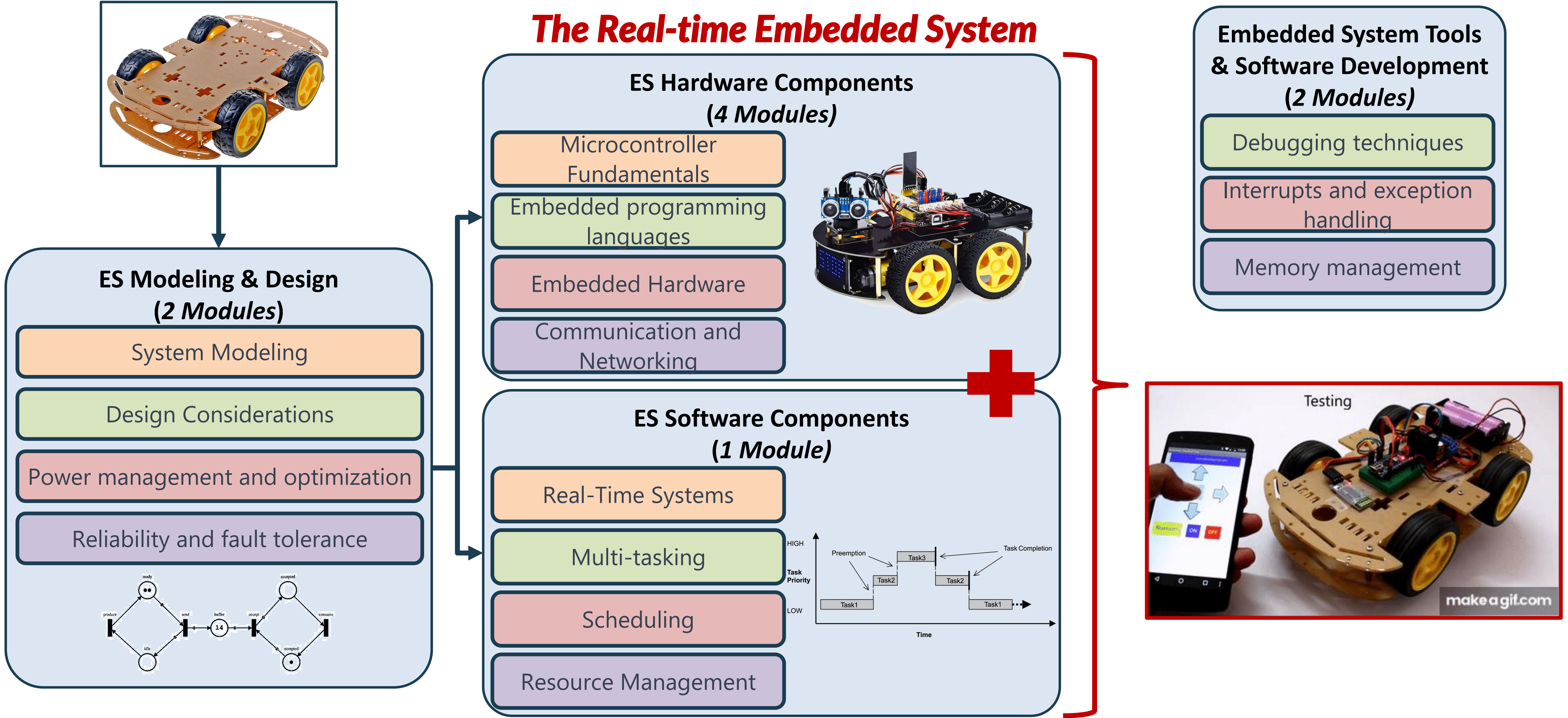
- This type of system can **never miss its deadline**.
- Missing the deadline may have **disastrous consequences**.
- The **usefulness** of results produced by a hard real-time system **decreases abruptly** and may become **negative if tardiness increases**.
- The **cost** of missing the deadline is often **loss of lives or/and huge damage**.




Types: *Weakly Hard* RT Systems

- A system in which the **distribution** of its **met** and **missed deadlines** during a window of time is precisely **bounded**.
- **Probabilistic guarantees** are sufficient for these systems





1. Wang, J. Real-time embedded systems. John Wiley & Sons, 2017.
2. Michael Barr and Anthony Massa. Programming embedded systems: with C and GNU development tools. " O'Reilly Media, Inc.", 2006.
3. Steve Heath. Embedded systems design. Elsevier, 2002.
4. Edward A Lee and Sanjit A Seshia. Introduction to embedded systems-a cyber-physical systems approach, 2011. URL <http://LeeSeshia.org>, 6:18.

For Further Inquiries, Please
 *send an email*

Catherine.elias@guc.edu.eg,
Catherine.elias@ieee.org

Thank you for your attention!

See you next time 😊