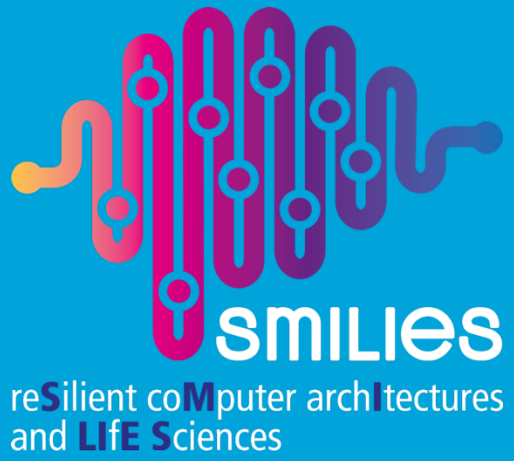




Politecnico di Torino

Department of Control and Computer Engineering

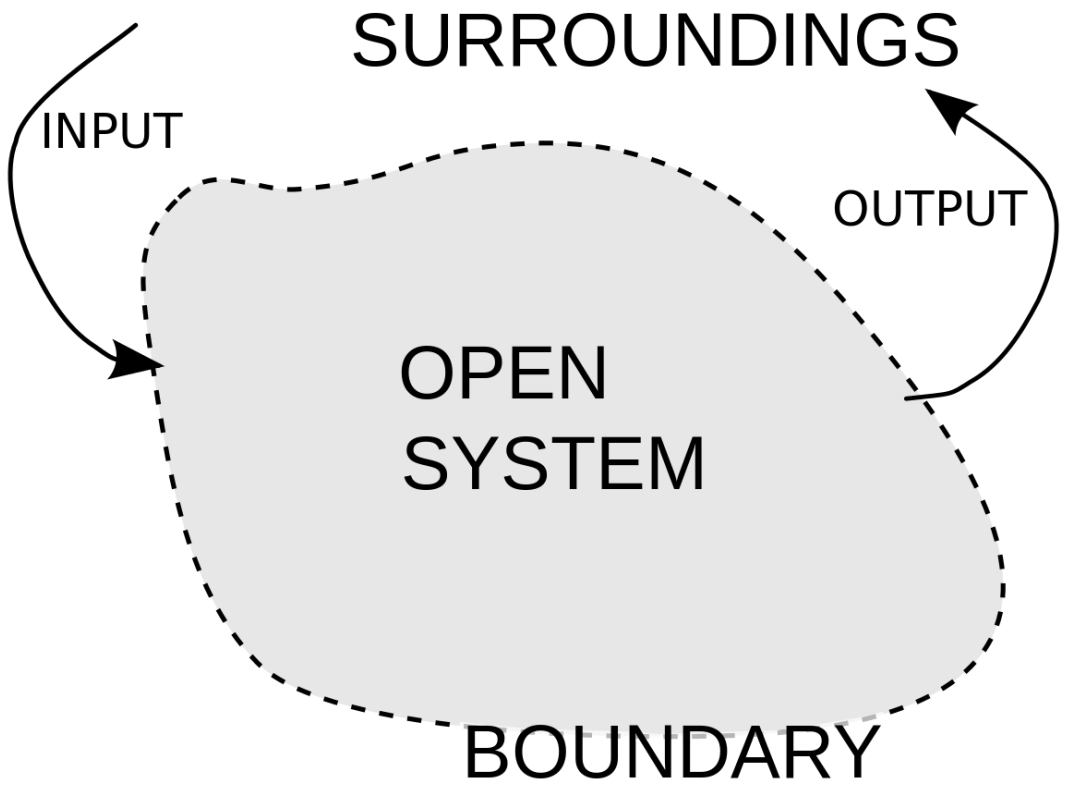

# EMBEDDED SYSTEMS INTRODUCTION

STEFANO DI CARLO

# WHAT IS A SYSTEM ?

- A **system is an arrangement** in which all its unit assemble **work together** according to a set of rules. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan.
- What about an **embedded syste**m?

**No unique definition!!!!**



# WHAT IS AN EMBEDDED SYSTEM ?

- An embedded systems is a **combination of hardware and software** where software is usually known as **firmware** that is embedded into the hardware
- An embedded system is a **microcontroller or microprocessor-based system** which is designed to **perform a specific task**
- An embedded system is **combination of computer hardware and software**, and perhaps **additional mechanical or other parts**, designed to perform a dedicated function. In some cases, embedded systems are part of a larger system or product, as is the case of an antilock braking system in a car

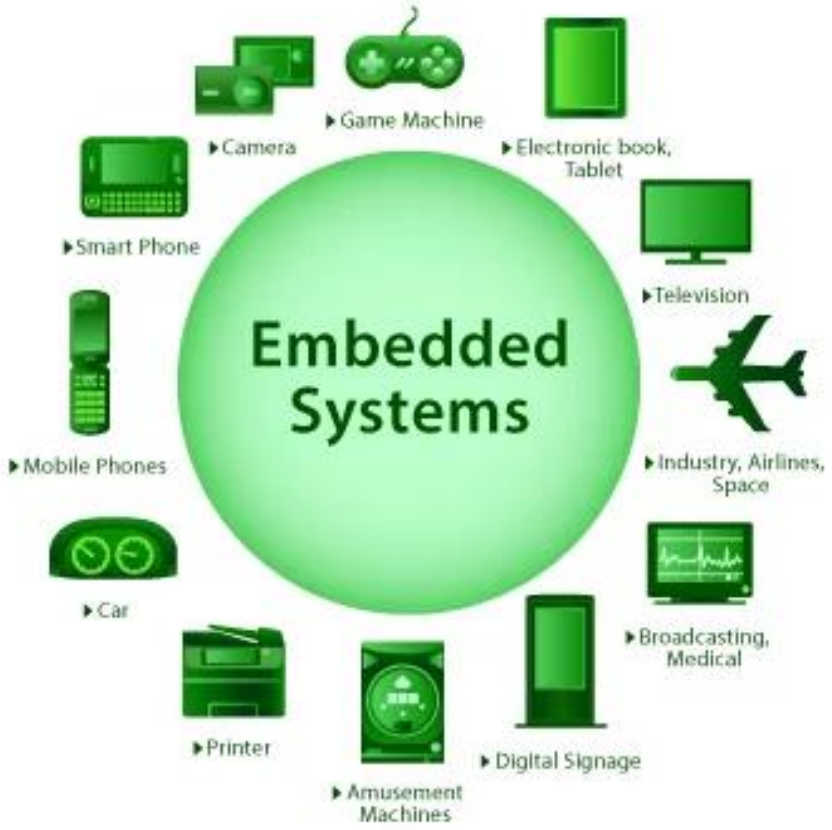
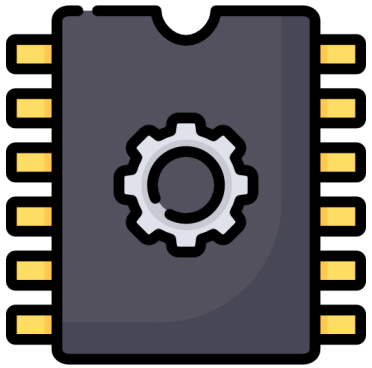


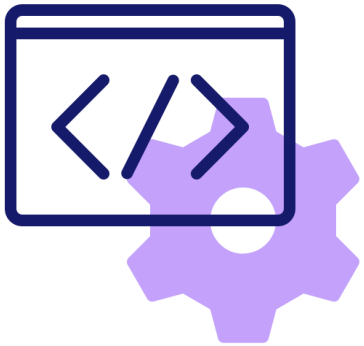

Image taken from https://www.bespokeroboticsautomation.com/embedded-systems/

# WHAT IS AN EMBEDDED SYSTEM ?

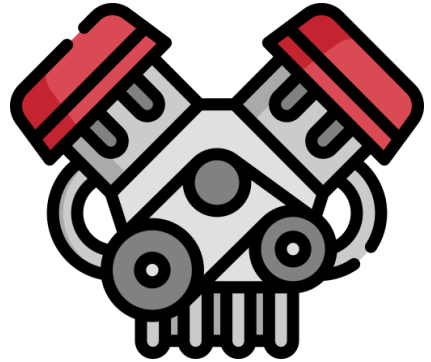
APPLICATIONS



HARDWARE



SOFTWARE



PHYSICAL SYSTEM

# CLASSIFICATION OF EMBEDDED SYSTEMS

- Based on functionality and performance requirements, embedded systems are classified as :
  - **Stand-alone**
  - **Real-time**
  - **Networked Information Appliances**
  - **Mobile/Wearable Devices**

# STAND-ALONE EMBEDDED SYSTEMS:

- Stand-alone systems operate independently, taking inputs, processing them, and producing the desired outputs
- Inputs can come from electrical signals (e.g., transducers) or user commands, such as pressing a button
- Outputs may include electrical signals that control other systems or visual feedback through LED or LCD displays
- These systems are common in various domains, including:
  - MP3 players
  - Digital cameras
  - Video game consoles
  - Microwave ovens
  - Temperature measurement systems

# REAL-TIME SYSTEMS

- Real-time systems are embedded systems that must complete specific tasks within a defined time period
- *Example*: *A system that must open a valve within 30 milliseconds when humidity exceeds a threshold. Failure to do so could lead to a catastrophe. Such systems are called* ***hard real-time systems****, as they have strict deadlines*
- In some cases, missing a deadline is less critical and doesn't cause catastrophic outcomes.
- Example: A DVD player that experiences a slight delay when responding to a remote control command. This delay is acceptable and doesn't have severe consequences. Such systems are called **soft real-time systems**

# NETWORKED INFORMATION APPLIANCES:

- An embedded system designed to connect to a network and exchange information, often for specific tasks
- Typically includes communication interfaces, sensors, actuators, and processors for dedicated functionalities
- They form the basis of the **Internet of Things (IoT)** that refers to the network of physical objects, or "things," embedded with sensors, software, and other technologies that enable them to collect and exchange data with other devices and systems over the internet
- These "things" can range from everyday household items like smart appliances to complex industrial machines

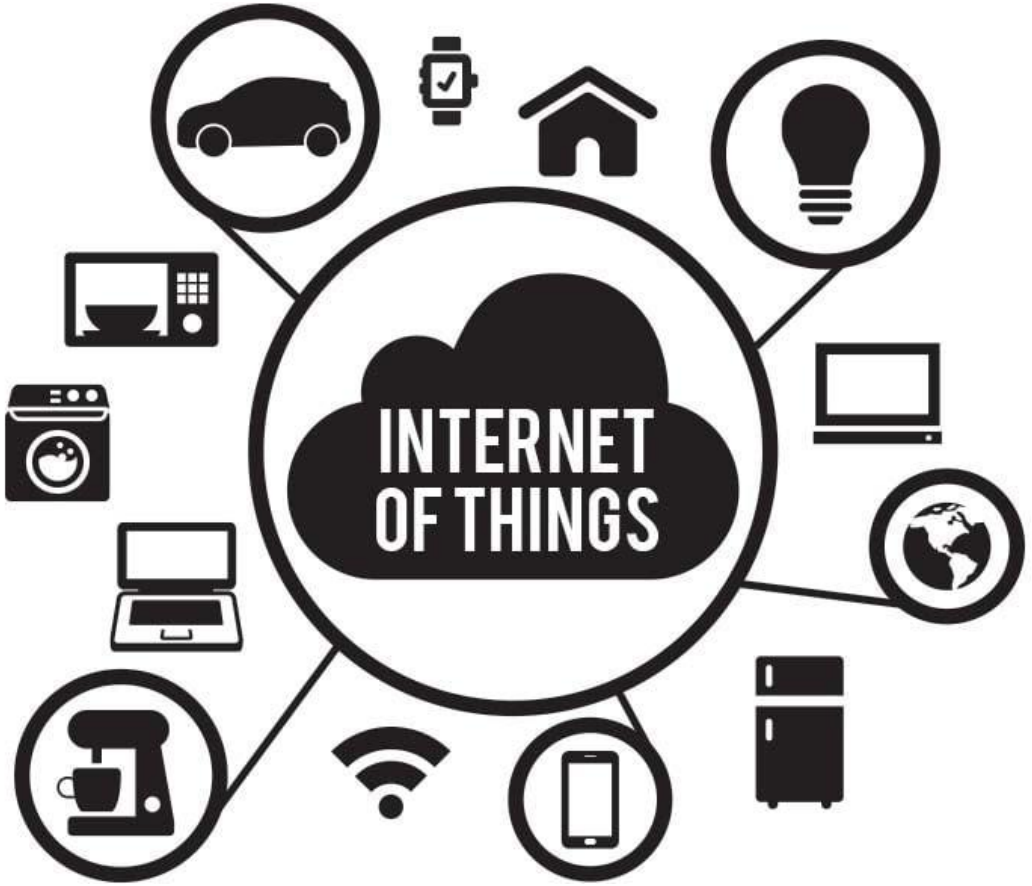



Image taken from https://thenewstack.io/what-does-it-mean-to-be-on-the-internet-of-things/-systems/

# MOBILE DEVICES

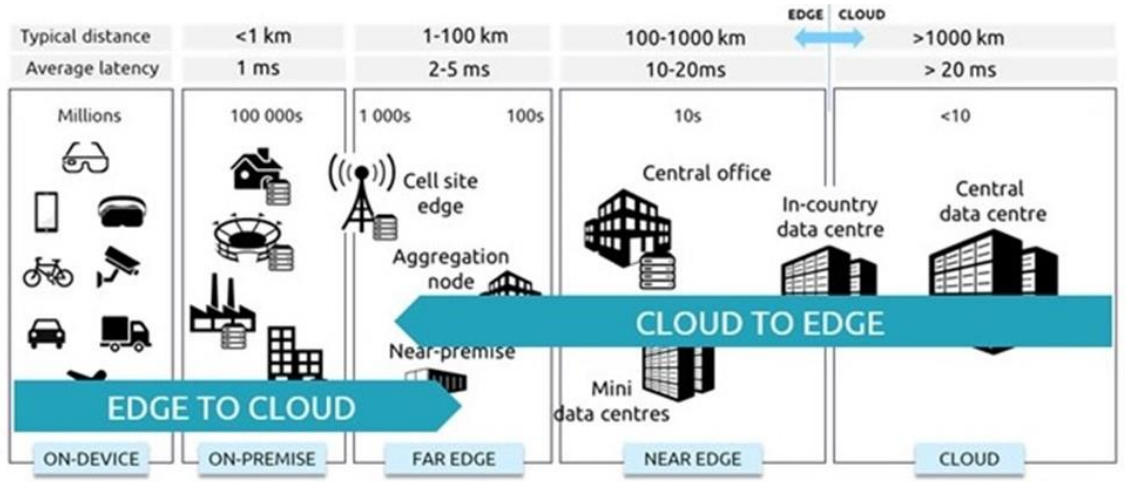

- Mobile devices are a special category of embedded systems
- Nowadays this category includes smartphones, tablets, wearables, smart home appliances using a large variety of technologies such as 5G, Wi-Fi, and AI technologies
- Though these devices do many general-purpose tasks, they need to be designed just like the ‘conventional’ embedded systems

# DIFFERENCE BETWEEN EMBEDDED AND GENERAL PURPOSE SYSTEM

- **Embedded System**
  - Designed to do some specific task
  - Not programmable by the end user
  - Low power consumption
  - Fixed time constraints
  - Specific tasks (that's why they are more economical)
- **General Purpose System**
  - Can perform multiple tasks
  - Programmable by the end user
  - Generally high-power consumption
  - Does not have fixed time constraints
  - More expensive that task specific systems

# THE COMPUTING CONITNUUM

- The progressive convergence between **Cloud Computing** and the Internet of Things (IoT) is resulting in a Computing Continuum
- The multi-faceted concept of Edge Computing first became to represent a middle ground between data centers and IoT hyper-local networks of sensors and actuators
- Then a much more nuanced paradigm emerged and placed computing infrastructure on a spectrum covering from the Cloud Data Centers to Edge Nodes with many intermediate levels
- High Performance Computing (HPC), Artificial Intelligence, 5G/6G networks are also part of this Continuum for which hardware and software need to be jointly considered.



# WHAT MAKES EMBEDDED SYSTEMS DIFFERENT?

**Real-Time Operation**: the system's ability to process data and deliver outputs within a defined time constraint.

**Size:** Size in embedded systems refers to the physical dimensions and footprint of the system. Many embedded systems are designed to be compact to fit within specific devices like smartphones, wearable technology, or medical implants. Reducing size without compromising functionality is often a key design constraint.

**Cost:** the expense of hardware components (such as processors, sensors, and memory) as well as development and production costs. Because embedded systems are often used in mass-produced devices, cost-efficiency is critical to keeping production viable.

**Time:** can refer to several aspects including development time, execution time, response time.


**Reliability:** ability to consistently perform its intended functions without failure. Since many embedded systems are used in critical environments (e.g., medical devices, automotive systems), they must maintain high reliability under varying conditions.

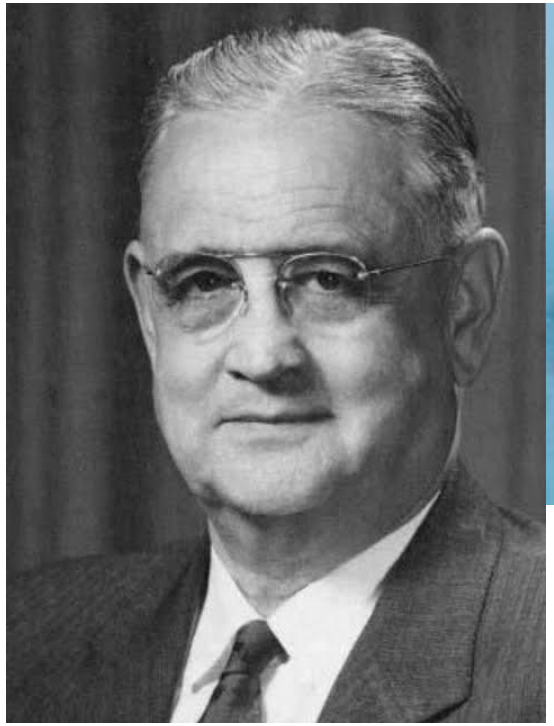

**Safety:** capability to operate without causing harm to the users or the environment. In safety-critical applications (e.g., aviation systems, industrial control), systems must comply with stringent safety standards and fail-safes must be implemented to prevent accidents or injuries.

**Energy**: the system's power consumption. Since many embedded systems are used in battery-operated devices (e.g., mobile phones, IoT sensors), low energy consumption is crucial to extend battery life and reduce operational costs. Designers often use power-saving techniques and energy-efficient processors to optimize energy usage.

# HISTORY OF EMBEDDED SYSTEMS

- In the earliest years of computers in 1930 – 40s, computers were sometimes dedicated to a single purpose task.
- One of the first recognizably modern embedded system was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory.

Image taken from https://en.wikipedia.org/wiki/Apollo_Guidance_Computer/

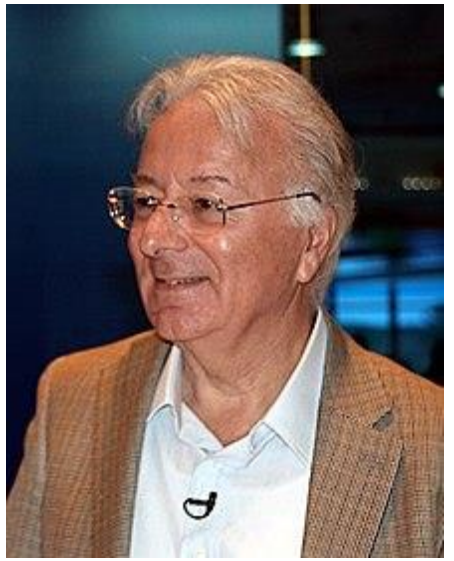


https://www.nmspacemuseum.org/inductee/charles-s-draper/

# HISTORY.....

- Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality.
- The first microprocessor for example, the Intel 4004 was designed for calculators and other small systems but still required many external memory and support chips.

Taken From https://en.wikipedia.org/wiki/Federico_Faggin



**Federico Faggini**

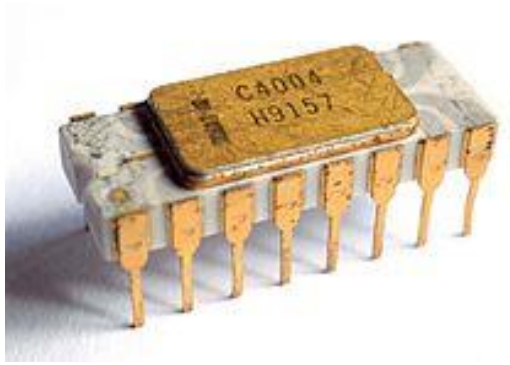
The father of the modern microprocessors designed the Intel 4004, the Zilog Z80 and several other CPUs.


Taken from :https://it.wikipedia.org/wiki/Motorola_6800


Taken from https://it.wikipedia.org/wiki/MOS_6502

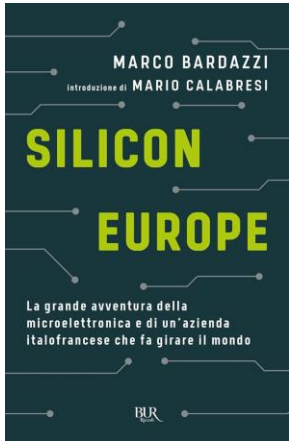
Taken From https://en.wikipedia.org/wiki/Zilog_Z80


Taken from https://en.wikipedia.org/wiki/Intel_4004



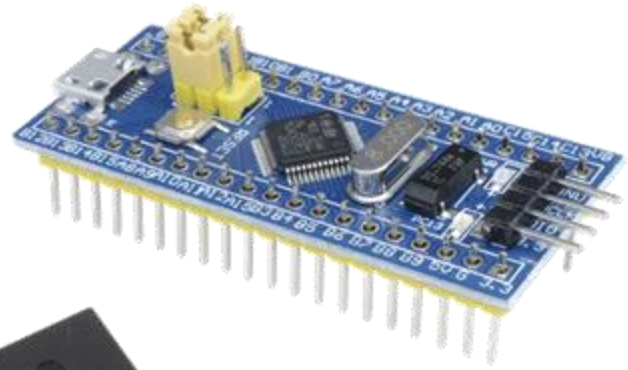
# HISTORYY.....!!!

- By the mid-1980s, most of the common previously external system components had been integrated into the same chip as the processor
- This modern form of the microcontroller allowed an even more widespread use, which by the end of the decade were the norm rather than the exception for almost all electronics devices.




**Silicon Europe**
Marco Bardazzi
Bocconi University Press, 2024
ISBN 9791280623232

**STM32F103C8T6**



**ATmega328**

**NXP LPC1768**

Taken from https://www.electronics-lab.com/top-10-popular-microcontrollers-among-makers

# PRESENT OF EMBEDDED SYSTEM.....

- Number of system per person was increased
- The population of embedded system outraced the population of people in the world


ARM

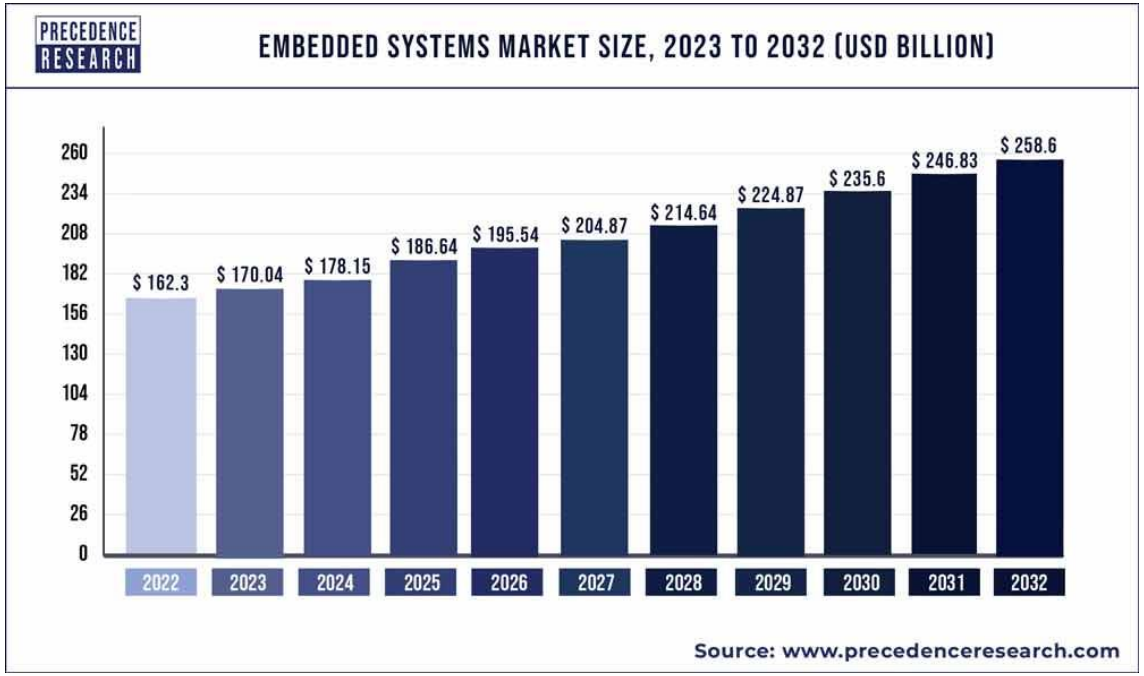
Taken from https://www.electronicsweekly.com/news/business/finance/microcontrollers-become-major-arm-2-2015-02/


RISC-V

Taken from https://www.digikey.it/it/articles/how-to-get-started-with-risc-v-based-microcontrollers

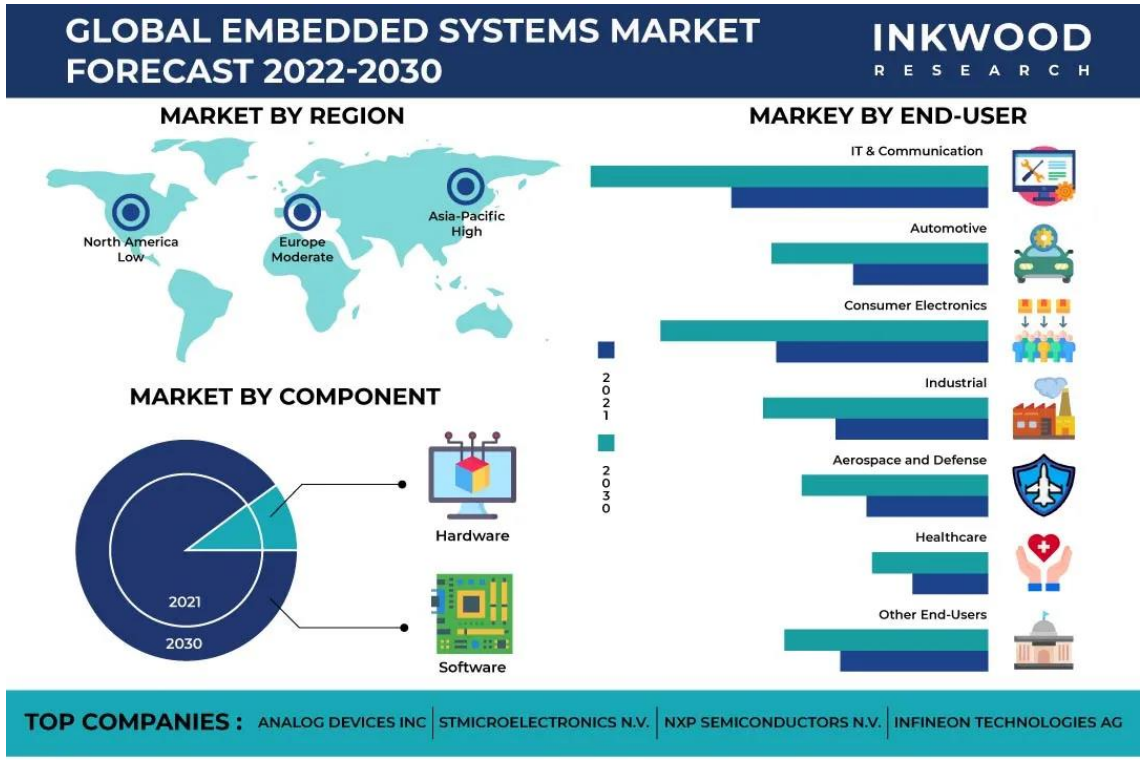
# EMBEDDED SYSTEMS MARKET SIZE AND GROWTH 2023 TO 2032

- The global embedded systems market size was reached at USD 162.3 billion in 2022 and is expected to hit around USD 258.6 billion by 2032, poised to grow at a CAGR of 4.77% during the forecast period from 2023 to 2032
- Key Takeaways:
  - North America has accounted for 51% revenue share in 2022.
  - Asia Pacific has held a 24% revenue share in 2022.
  - By function, the standalone system segment has held a 69% revenue share in 2022.
  - The mobile system segment has generated a revenue share of around 13% in 2022.
  - By hardware, ASIC & ASSP segment has generated a 31.5% revenue share in 2022.
  - The microprocessor segment has captured a 22.3% revenue share in 2022.
  - By Application, the manufacturing segment has accounted for 11% of revenue share in 2022.




Taken from https://www.precedenceresearch.com/embedded-systems-market

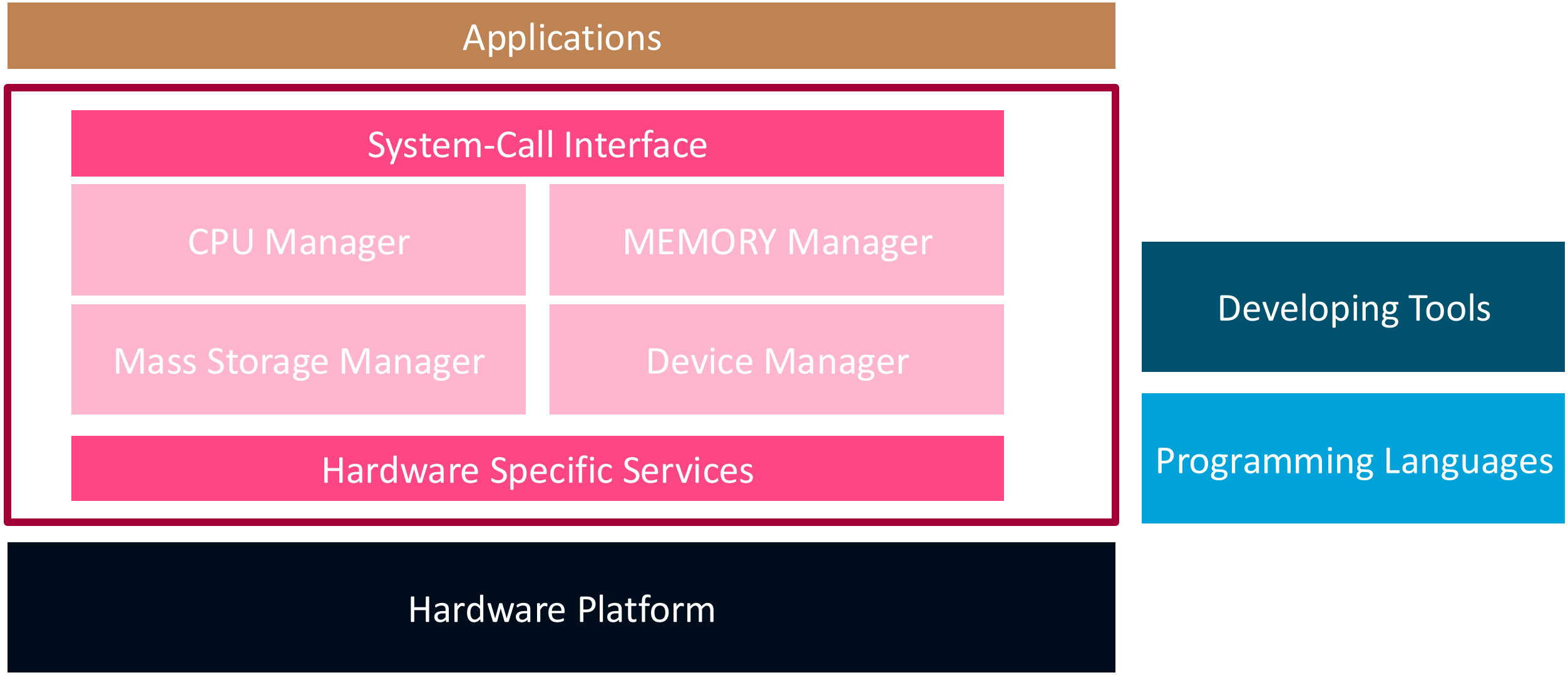
# EMBEDDED SYSTEMS MARKET SIZE AND GROWTH 2023 TO 2032



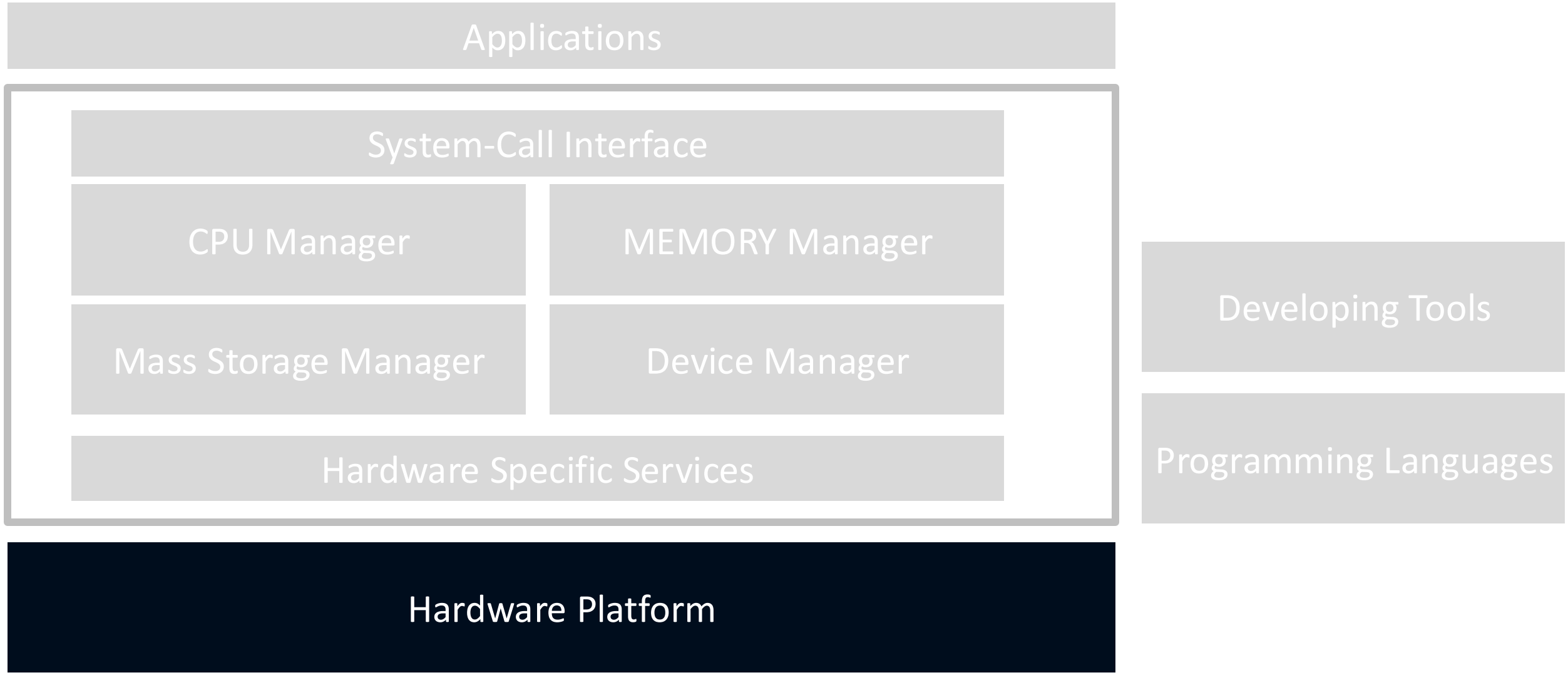

Taken from https://www.inkwoodresearch.com/reports/embedded-systems-market/

- Global Embedded Systems Market by:
  - Component:
    - Hardware: Processor IP, MPU/MCU, RAM, Flash Memory, DSP, ASIC, Programmable Logic Devices, Embedded Boards)
    - Software: Operating Systems, Software Development and Testing Tools, Middleware, Embedded Linux))
  - End-user: IT & Communication, Automotive, Consumer Electronics, Industrial, Aerospace and Defense, Healthcare, Other End-users
  - Geography

# EMBEDDED SYSTEMS DEVELOPMENT PLATFORM

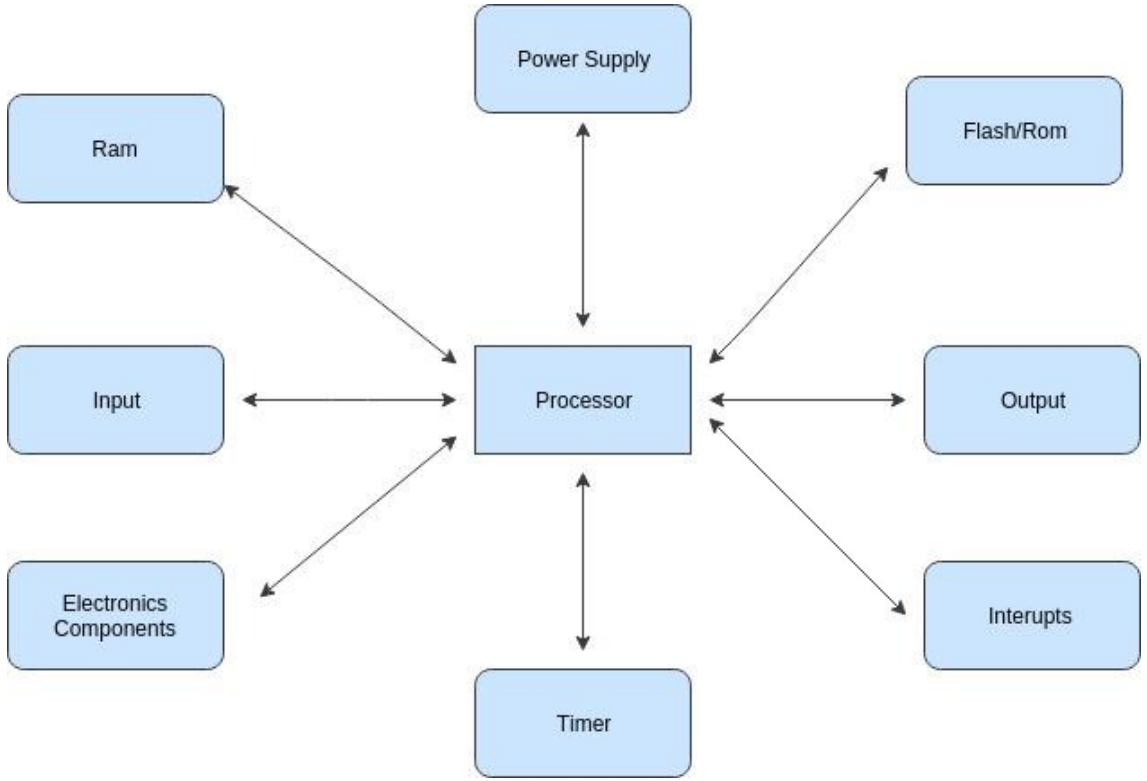


# EMBEDDED SYSTEMS DEVELOPMENT PLATFORM



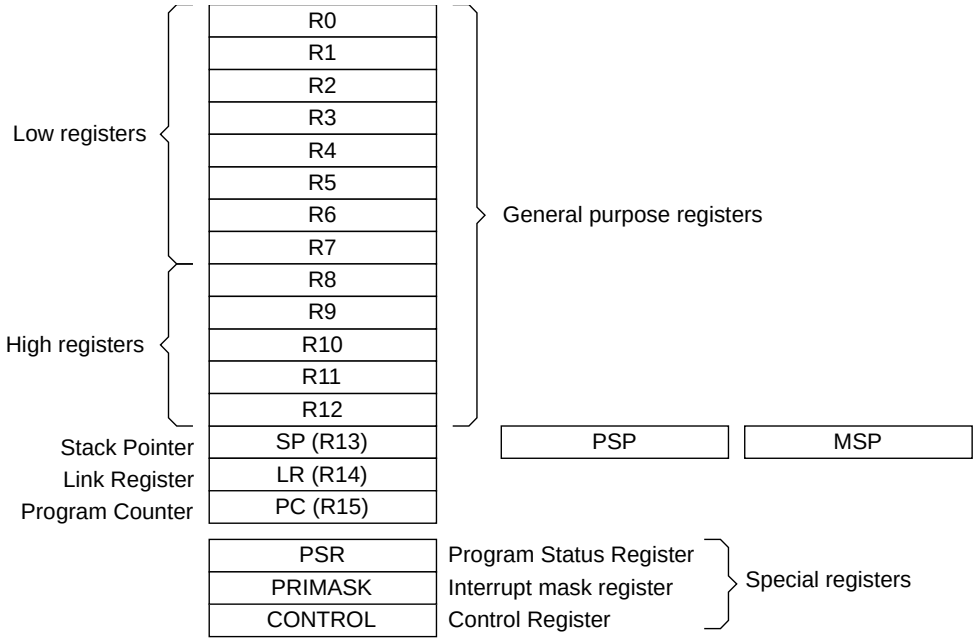
# HARDWARE PLATFORM

- An embedded system uses a hardware platform to perform the operation.
- Hardware of the embedded system is assembled with a microprocessor/ microcontroller.
- It has the elements such as input/ output interfaces, memory, user interface and the display unit.

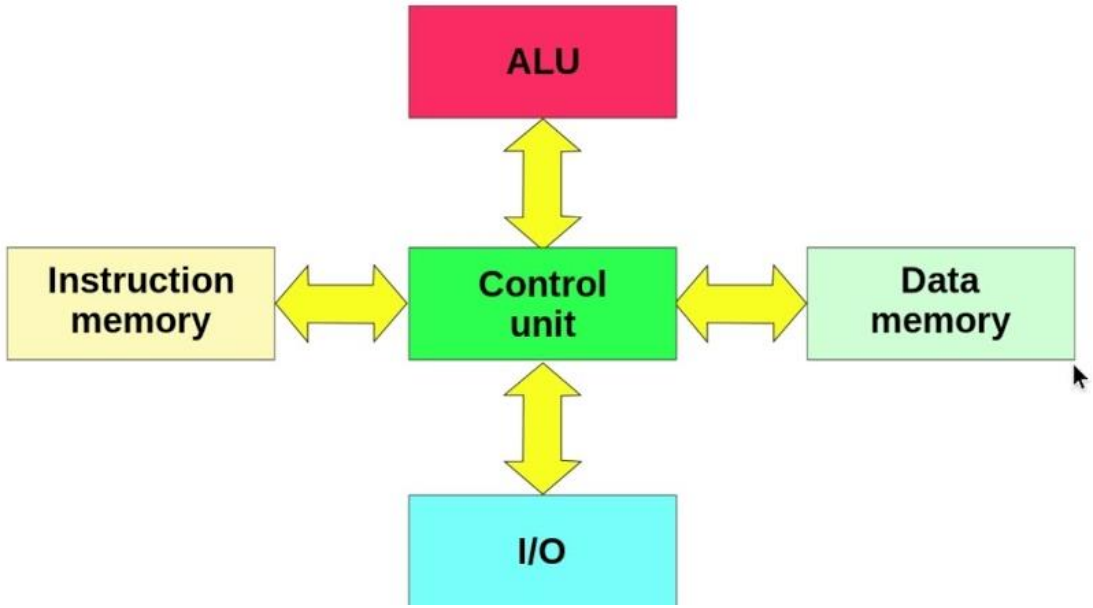


# THE CPU AT THE CORE OF AN EMBEDDED SYSTEM

- The **Central Processing Unit (CPU)** is the core component of any embedded system, responsible for executing instructions and performing tasks.
- Any CPU always performs and endless loop:
  - Fetch one instructions from memory (read cycle)
  - Execute the instruction
  - Repeat from the beginning
- The CPU contains memory elements to operate:
  - Program Counter (PC) or Instruction pointer (IP) to store the address of the next instruction to be executed
  - Register File (RF) to store results of computations
  - Processor status word (PSW) to store CPU status (configuration bits, more later), ….



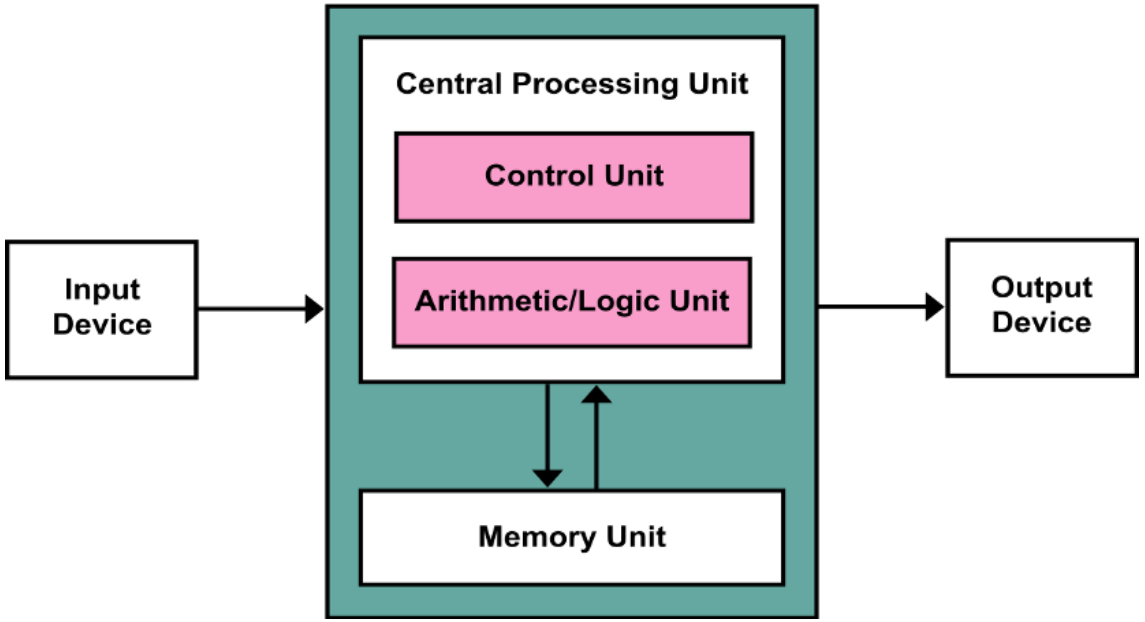
# THE HARVARD ARCHITECTURE




Often used in microcontrollers and digital signal processors (DSPs), where fast, efficient processing is needed, such as in real-time embedded systems.

- A computer architecture that **separates** the storage and handling of instructions and data into two distinct memory spaces:
  - **Instruction Memory**: Stores the program code or instructions that the CPU will execute.
  - **Data Memory**: Stores the data that the program operates on.
- Key Features:
  - **Parallel Access:** Since instruction and data memories are separate, the CPU can fetch an instruction and read/write data simultaneously, improving performance.
  - **Different Memory Sizes:** Instruction memory and data memory can have different sizes and speeds, optimized for their specific roles.

# THE VON NEUMANN ARCHITECTURE




The Von Neumann architecture is commonly used in general-purpose computers, including personal computers and servers, due to its flexibility and simplicity in design.

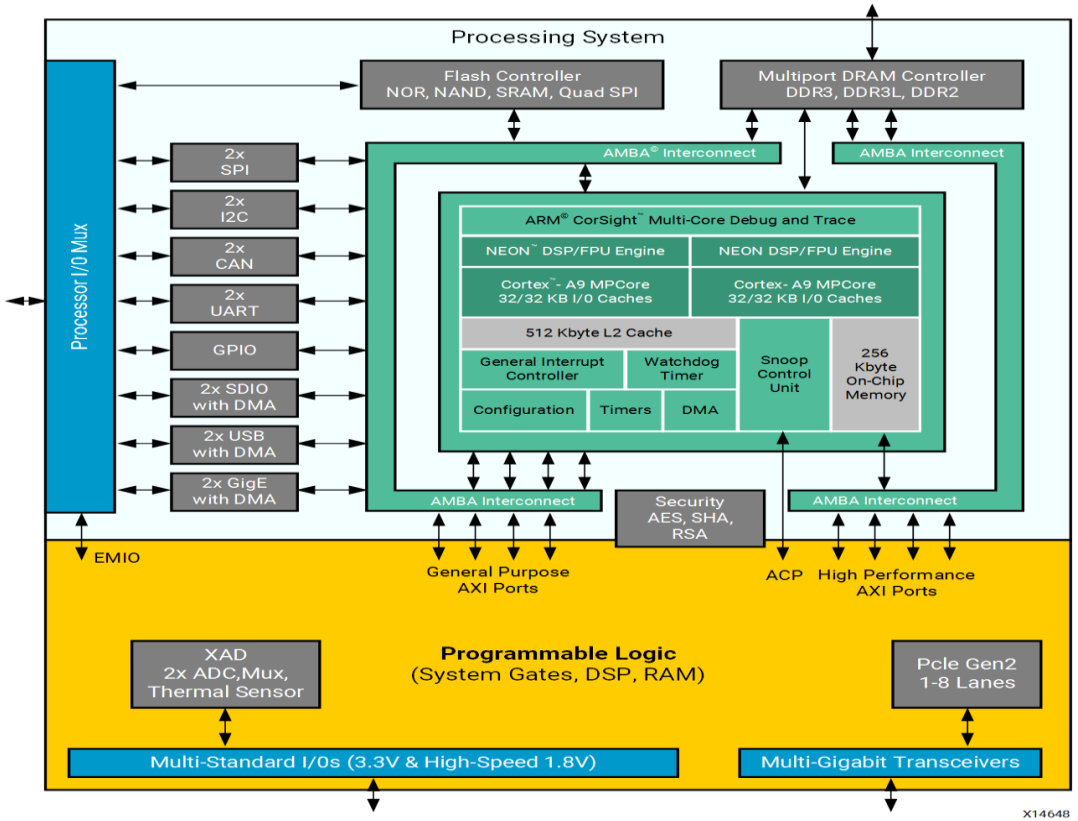
- A computer architecture model consisting of a single shared memory for programs and data accessed by the CPU.
- It is named after the mathematician John von Neumann, who proposed this design in the 1940s.
- Key Features:
  - **Single Memory Space**: Both instructions (program code) and data are stored in the same memory space. This simplifies the design.
  - **Sequential Execution:** The CPU fetches instructions one at a time, processes them, and then fetches the next one. This sequential operation can lead to a bottleneck, known as the "Von Neumann bottleneck," where the speed of processing is limited by the memory speed.
  - **Shared Data and Instruction Path**: The architecture uses a shared pathway (bus) for both data and instructions, meaning that during a read or write operation, the CPU cannot simultaneously fetch instructions and data.

# MICROPROCESSORS VS. MICROCONTROLLERS

- CPU in Microprocessors and System-on-Chip (SoC)
  - A Microprocessor contains the CPU as a standalone component.
  - It needs external components (like memory, I/O, and timers) to function.
  - Often part of a System-on-Chip (SoC), which integrates the CPU with memory, I/O, and other peripherals into one chip.
  - Example: Processors used in mobile phones and tablets.
- CPU in Microcontrollers
  - A Microcontroller (MCU) integrates the CPU along with memory (RAM, ROM) and I/O peripherals into one compact, low-power unit.
  - Designed for specific, small-scale tasks, offering real-time control.
  - Example: MCUs used in home appliances, automotive systems, and IoT devices.

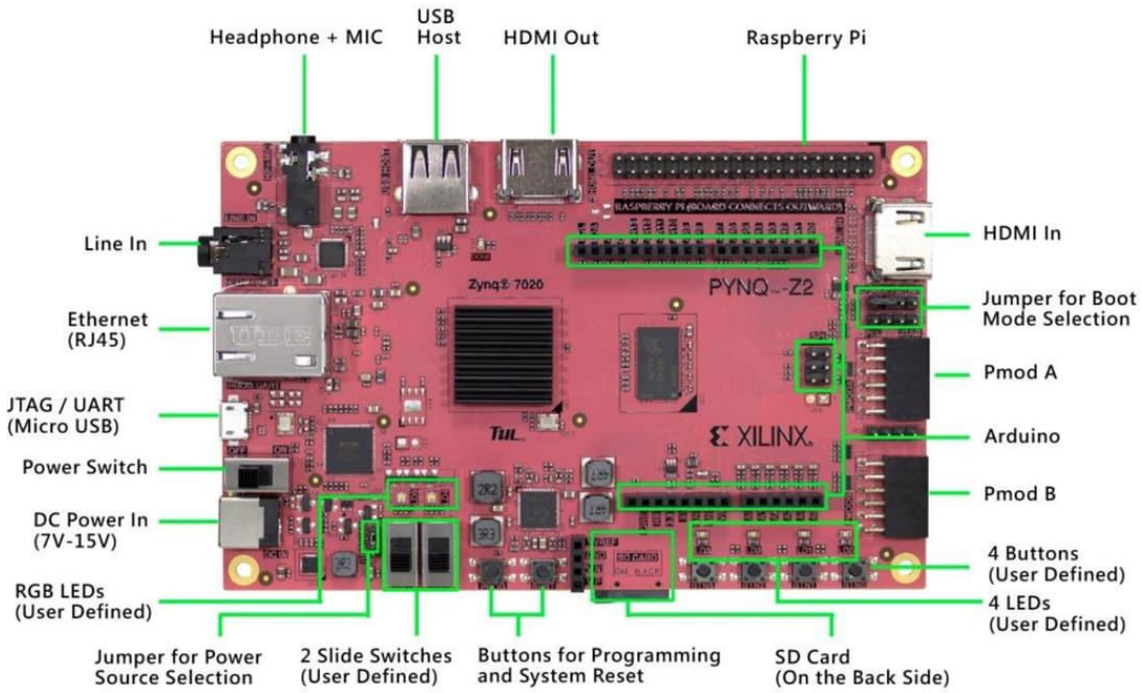
# THE XILINX ZYNQ 7000 SOC FAMILY

- The **Zynq-7000 family** of SoCs addresses high-end embedded-system applications, such as video surveillance, automotive-driver assistance, next-generation wireless, and factory automation.
- Zynq-7000 integrates a complete **ARM Cortex-A9** MPCore processor based 28nm system.
- For software developers, Zynq-7000 appears the same as a standard, fully featured ARM processor-based System-on-Chip (SoC), booting immediately at power-up and capable of running a variety of operating systems independently of the programmable logic.




Taken from https://xilinx.github.io/Embedded-Design-Tutorials/docs/2021.2/build/html/docs/User_Guides/SPA-UG/docs/1-introduction.html

# THE PYNQ DEVELOPMENT BOARD




Taken from https://www.mouser.it/new/dfrobot/dfrobot-pynqz2-dev-board/

- The **PYNQ-Z2** board is a development platform designed for prototyping and developing applications with Xilinx's Zynq System-on-Chip (SoC) architecture.
- Key Features:
  - **Zynq SoC**: The board integrates a Xilinx Zynq-7000 series FPGA, which combines a powerful ARM Cortex-A9 dual-core processor with programmable logic (FPGA) in a single device.
  - **Memory**: It typically includes 512 MB of DDR3 RAM, providing ample memory for running applications and processing data.
  - **Connectivity**: The PYNQ-Z2 offers various connectivity options, including HDMI input/output, USB ports, Ethernet for network connectivity, and a GPIO header for interfacing with external devices.
  - **Compatible with PYNQ Framework**: The board supports the PYNQ (Python Productivity for Zynq) framework, allowing developers to use Python to design and deploy applications easily, making it accessible for those familiar with high-level programming.
  - **Peripheral Interfaces**: It includes multiple peripheral interfaces such as PMOD connectors for adding additional hardware modules, making it versatile for various applications.

# STM32F446RE MICROCONTROLLER

- Based on the high-performance **Arm® Cortex®-M4 32-bit RISC core** operating at a frequency of up to 180 MHz.
- Incorporates high-speed embedded memories (Flash memory up to 512 Kbytes, up to 128 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix.
- It offers three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers.
- It features standard and advanced communication interfaces.


https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html

# STM32 NUCLEO-64 DEVELOPMENT BOARD

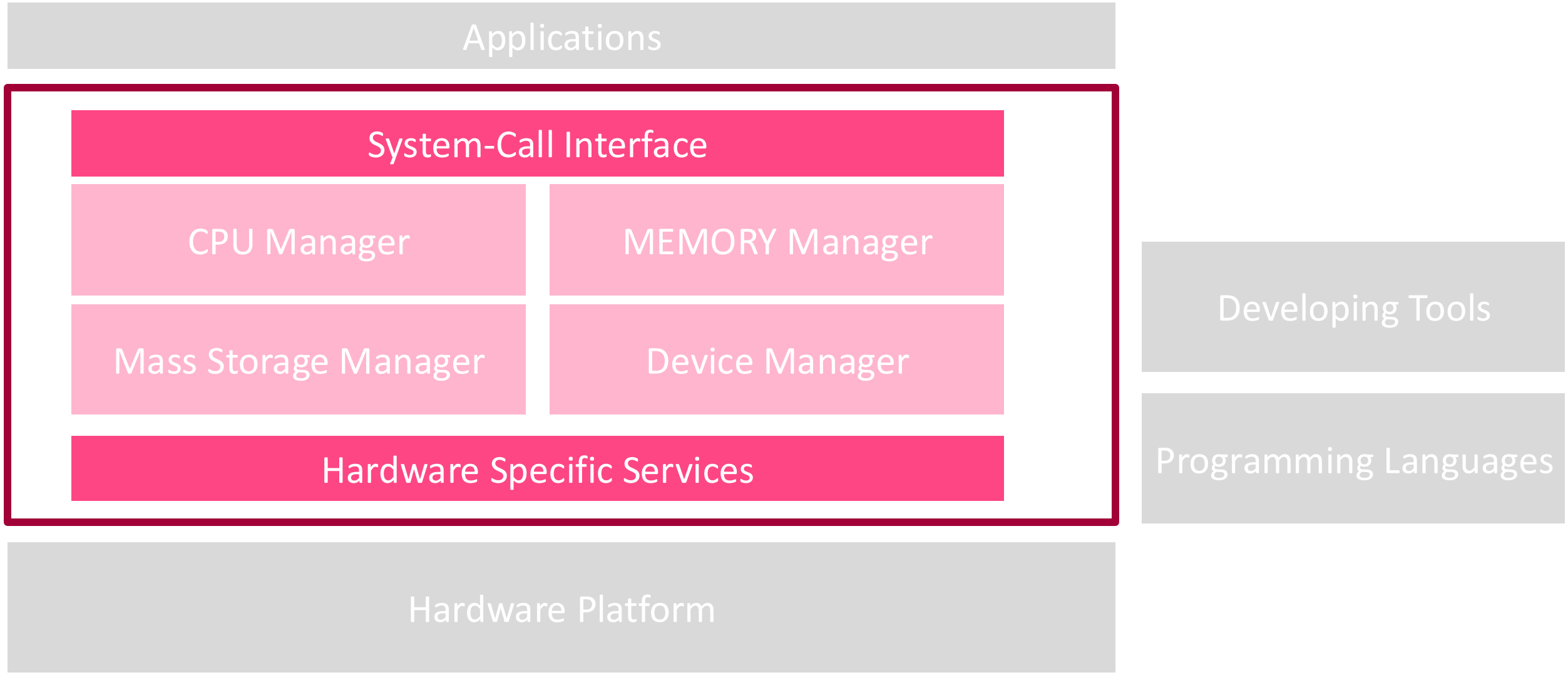
- The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features provided by the STM32 microcontroller
- Key features
  - STM32 microcontroller in an LQFP64 or LQFP48 package
  - 1 user LED
  - 1 user and 1 reset push-buttons
  - 32.768 kHz crystal oscillator
  - Board connectors:
  - ARDUINO® Uno V3 expansion connector
  - ST morpho extension pin headers for full access to all STM32 I/Os
  - Flexible power-supply options: ST-LINK USB VBUS or external sources
  - Comprehensive free software libraries and examples available with the STM32Cube MCU Package
  - Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE


https://www.st.com/en/evaluation-tools/nucleo-f446re.html

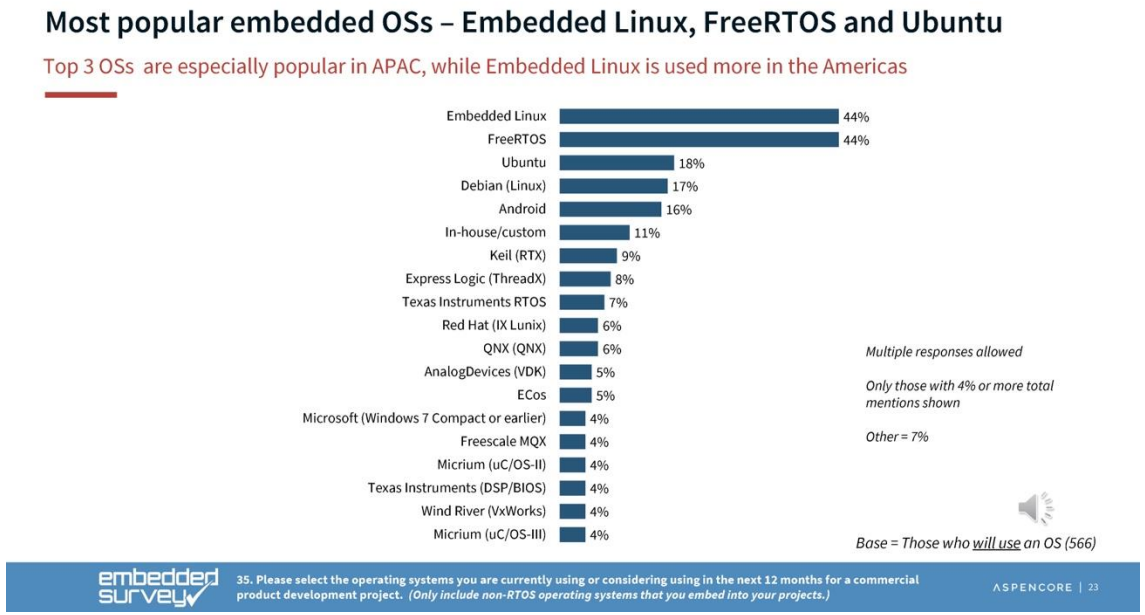
# DIFFERENCE BETWEEN MICROPROCESSOR AND MICROCONTROLLER

- MICROPROCESSOR
  - Microprocessor assimilates the function of a central processing unit (CPU) on a single integrated circuit (IC)
  - CPU needs external components
  - Typically, more powerful
  - The clock frequency is very high usually in the order of giga Hertz
  - Instruction throughout is given higher priority than interrupt latency
  - Used in complex systems
- MICROCONTROLLER
  - A microcontroller can be considered as a small computer that has a processor and some other components to make it a computer
  - Microcontrollers are generally used in embedded systems.
  - Typically, more energy-efficient
  - The clock frequency is less usually in the order of Megahertz.
  - In contrast, microcontrollers are designed to optimize interrupt latency.
  - Used in single-purpose applications

# EMBEDDED SYSTEMS DEVELOPMENT PLATFORM



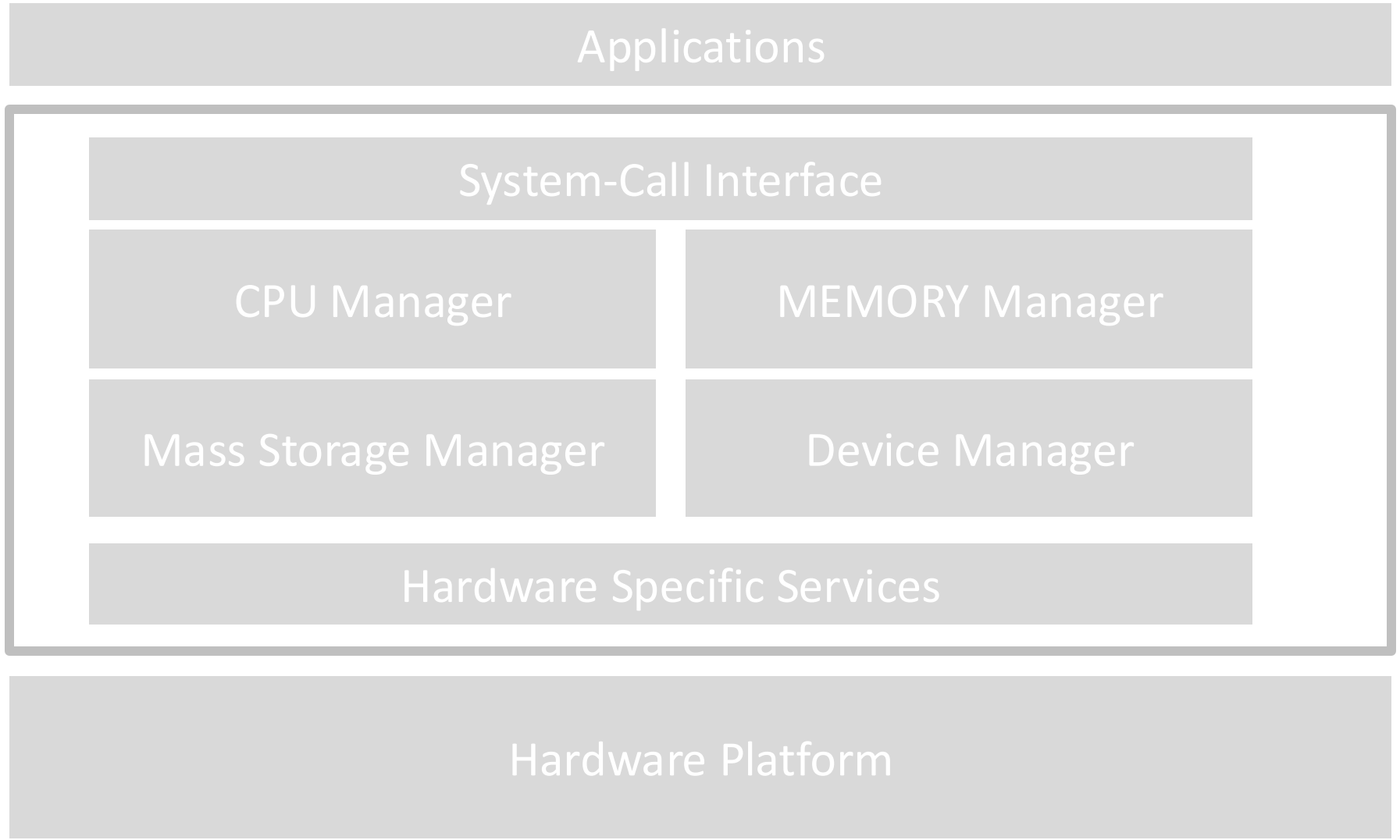
# WHICH OS FOR AN EMBEDDED SYSTEM?




Taken from https://www.embedded.com/embedded-survey-2023-more-ip-reuse-as-workloads-surge/

- Most embedded projects depend on an operating system in some form
- **Embedded Linux** and **FreeRTOS** head the list, followed by Ubuntu, Debian, Android, and RTX, and then ThreadX.
- Dependence on open-source or custom proprietary solutions continues to predominate;
  - only four in ten projects use a commercial OS either in whole or in part.
  - the reasons for this are that development teams wish to avoid relying on commercial suppliers for reasons related to cost, ease of use, compatibility, security and so forth, in addition to simply being satisfied with what they have now

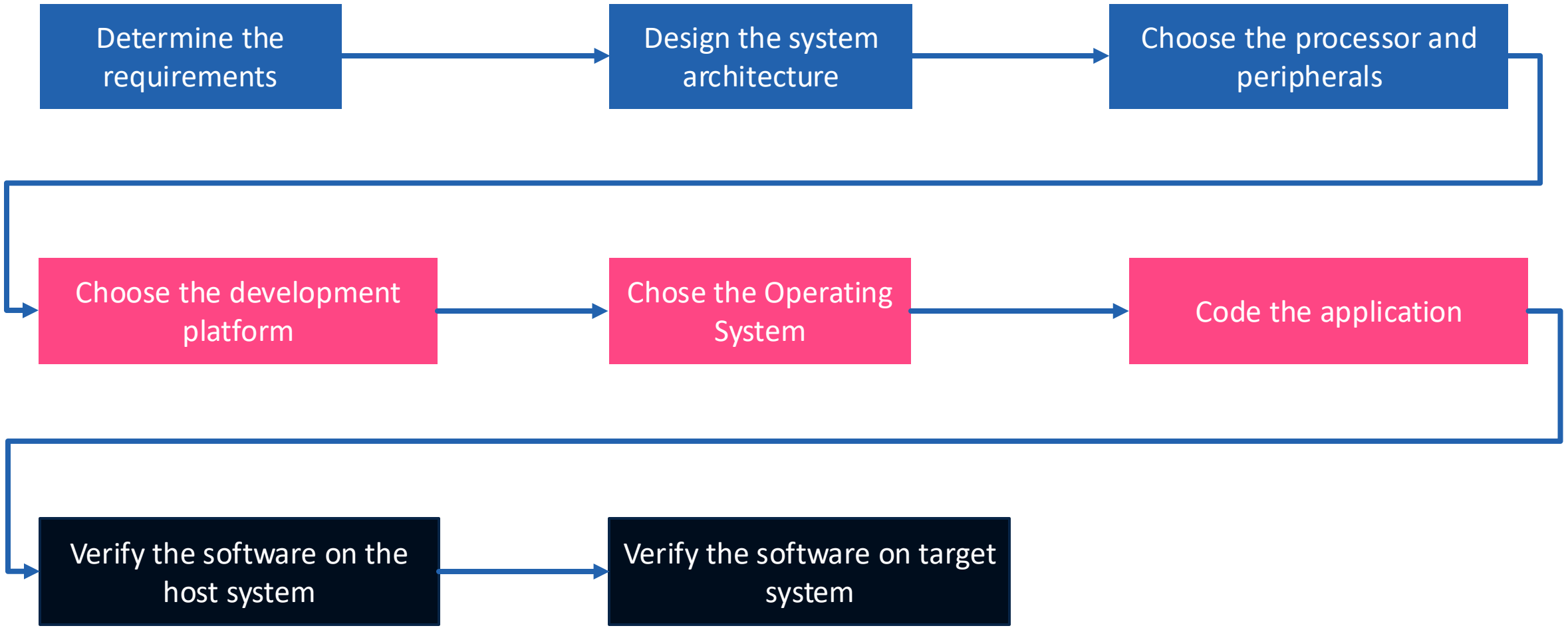
# EMBEDDED SYSTEMS DEVELOPMENT PLATFORM




Developing Tools

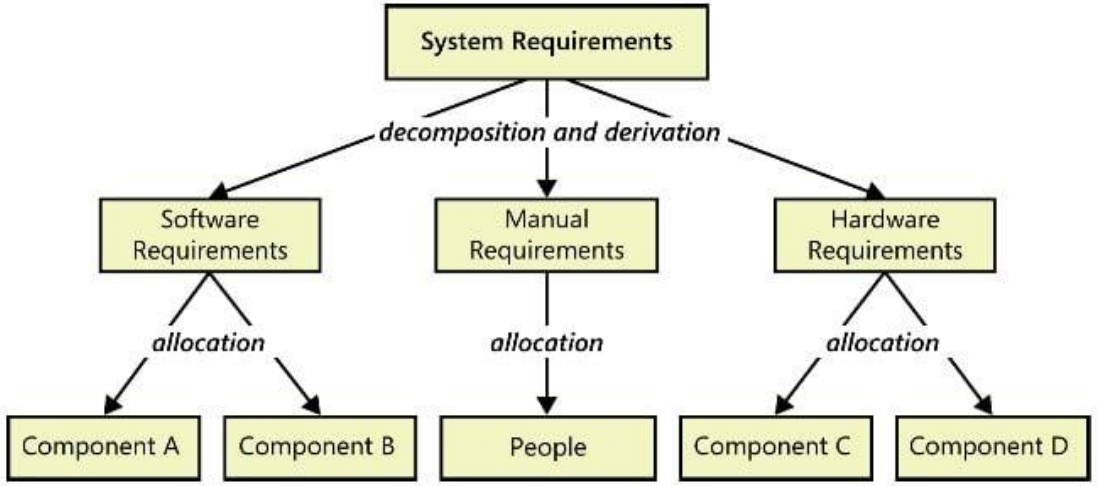
Programming Languages

# EMBEDDED SYSTEM DESIGN PROCESS



# REQUIREMENTS

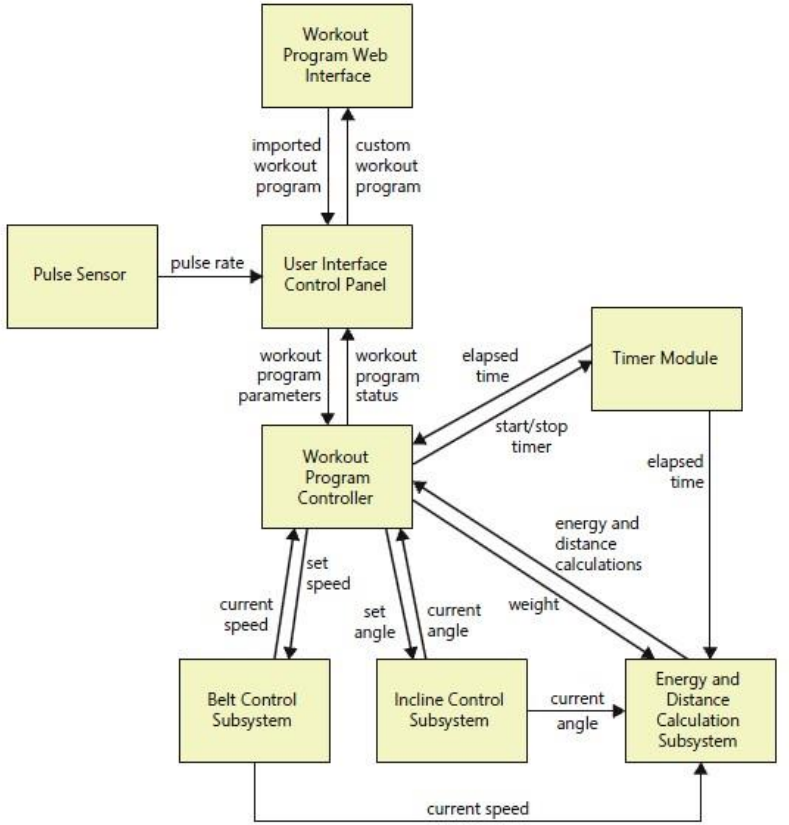
- Functional and non-functional.
  - Multi function or Multi mode system.
  - Size, cost, Weight etc.
- Selecting the H/W components.
  - Application specific H/W.
  - External interfaces.
  - Input devices.
  - Output devices.




Taken from https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3150/Requirements-for-Devices-Around-Us-Embedded-Systems.aspx

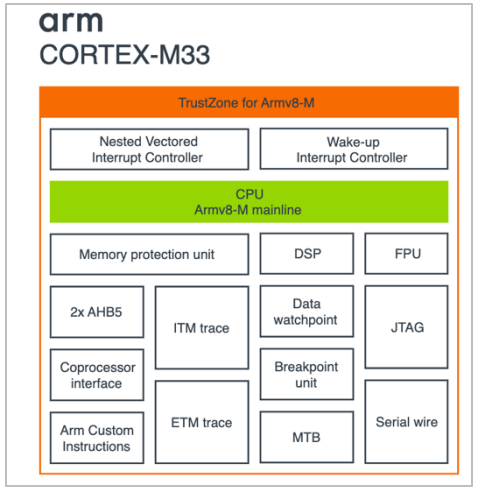
# DESIGN SYSTEM ARCHITECTURE

- System architecture depends on,
  - Whether the system is real-time.
  - Whether OS needs to be embedded.
  - Size, Cost, Power consumption etc.
- Visual modeling is a powerful analysis technique for specifying embedded systems.
- One useful model is an architecture diagram.




Taken from https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3150/Requirements-for-Devices-Around-Us-Embedded-Systems.aspx

# CHOOSE THE CPU AND DEVELOPMENT BOARD




Taken from https://developer.arm.com/Processors/Cortex-M33

https://www.st.com/en/evaluation-tools/nucleo-f446re.html

- Analyze the application requirements:
  - Processing Power: Choose single-core or multi-core based on application complexity.
  - Memory: Assess RAM and storage needs.
- Evaluate Peripheral Support:
  - I/O Capabilities: Ensure adequate ports (GPIO, UART, etc.) for connections.
  - Connectivity: Check for built-in options (Ethernet, Wi-Fi, Bluetooth).
- Assess Power Consumption:
  - Energy Efficiency: Analyze power needs, especially for battery-operated systems.
  - Thermal Management: Consider heat dissipation requirements.
- Consider Development Ecosystem:
  - Toolchain Support: Ensure compatibility with preferred development tools.
  - Community & Documentation: Choose platforms with strong support and resources.

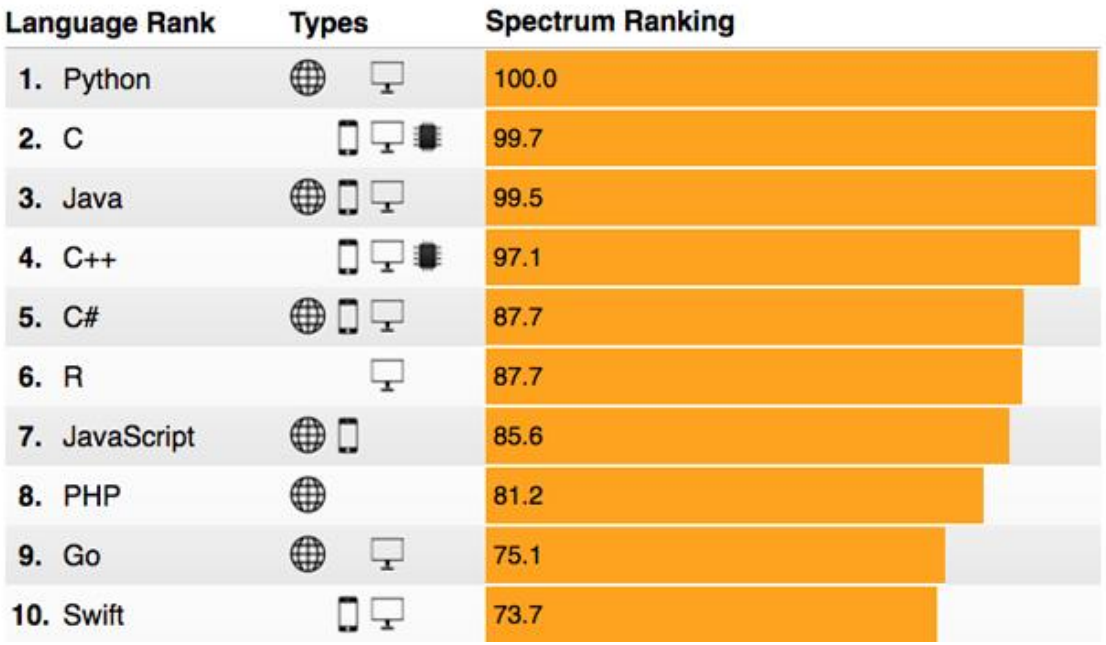
# CHOOSE OPERATING SYSTEM

- Analyze the application requirements:
  - Applications like robotics, automotive systems, and industrial control require **RTOS** for predictable timing (e.g., FreeRTOS, Micrium-OS, etc.)
  - Applications such as consumer electronics, data processing, often work effectively with **non-real-time OS** such as embedded Linux
- Assess Compatibility:
  - **Hardware Support:** Ensure the OS supports the target hardware and peripherals.
  - **Third-Party Libraries:** Check availability of libraries and frameworks for development.


Taken from https://www.pertech.co.il/embedded-wizard/

# CODE THE APPLICATION

- Choice of language.
  - Assembly.
  - C language.
  - Object Oriented Languages
    - (C++, Java etc.).
- Optimize the code



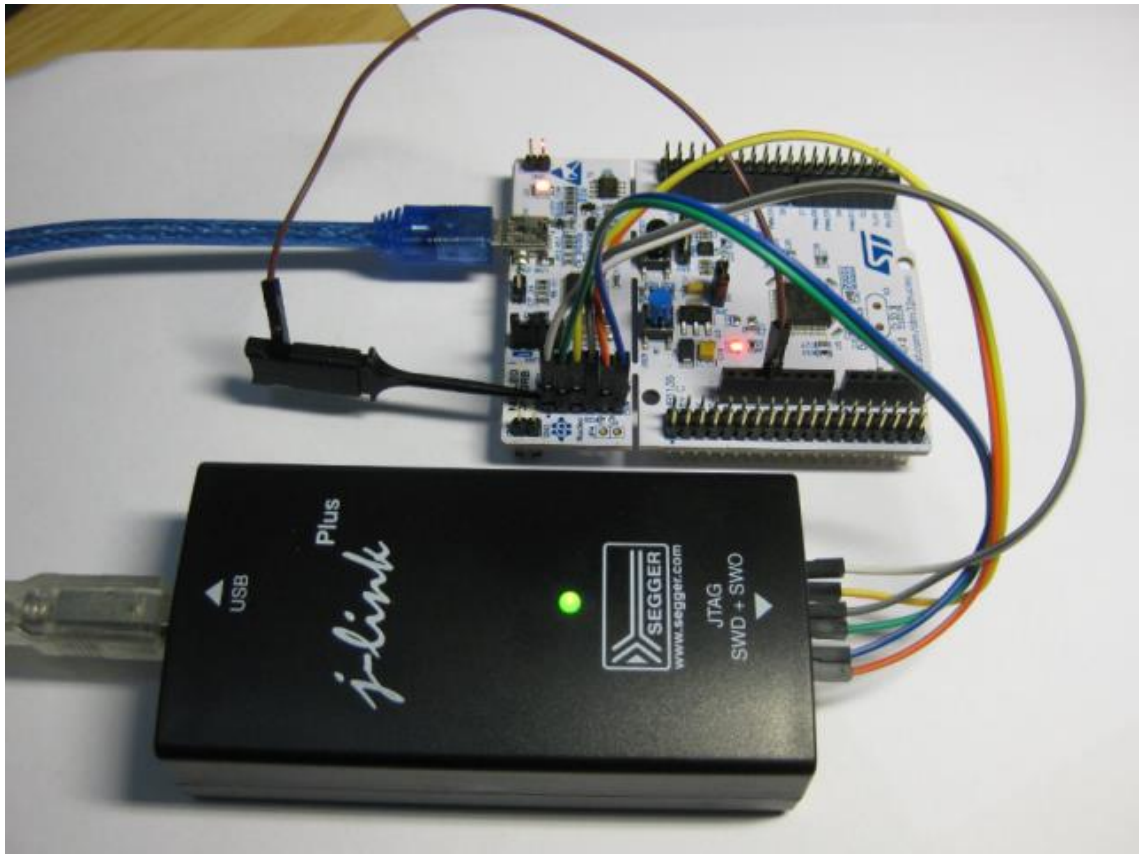
# VERIFY SOFTWARE ON THE HOST SYSTEM

- Compile and assemble the source code into object file.
- Use a simulator to simulate the working of the system.
- We will use QEMU, an **open-source emulator** and virtualizer that allows users to **run operating systems and programs for one architecture on another**, providing a flexible and efficient platform for system development and testing.


https://www.freertos.org/Documentation/02-Kernel/03-Supported-devices/04-Demos/Texas-Instruments/cortex-m3-qemu-lm3S6965-demo

# VERIFY SOFTWARE ON TARGET SYSTEM

- Transfer the compiled application to the target device using an appropriate programmer interface (e.g., JTAG interface)
- Execute the software on the target system. Monitor outputs through serial communication or debugging interfaces
- Validate the software functionality against requirements, using test cases to ensure all features work as intended
- Review logs, console outputs, and behavior to identify any issues. Adjust the code as needed.



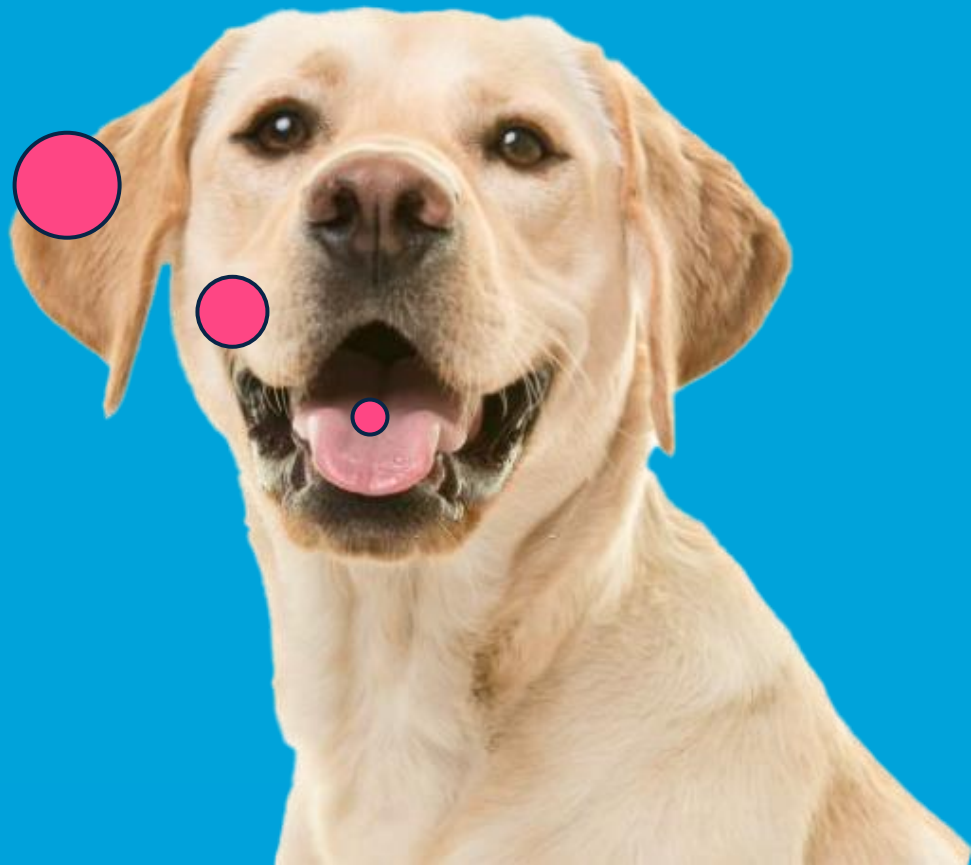

https://mcuoneclipse.com/2015/08/22/debugging-stm32f103rb-nucleo-board-with-with-segger-j-link/

SMILIES
reSilient coMputer archItectures and LIfE Sciences

Politecnico di Torino
Department of Control and Computer Engineering

DAUIN

QUESTIONS?



# THANK YOU!