

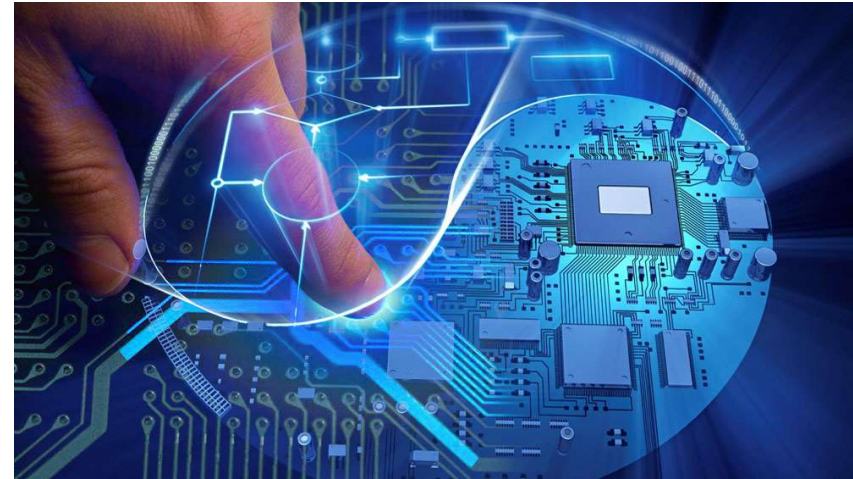
Introduction to Embedded Systems

Tien Pham Van, Dr. rer. nat.

Hanoi University of Science and Technology

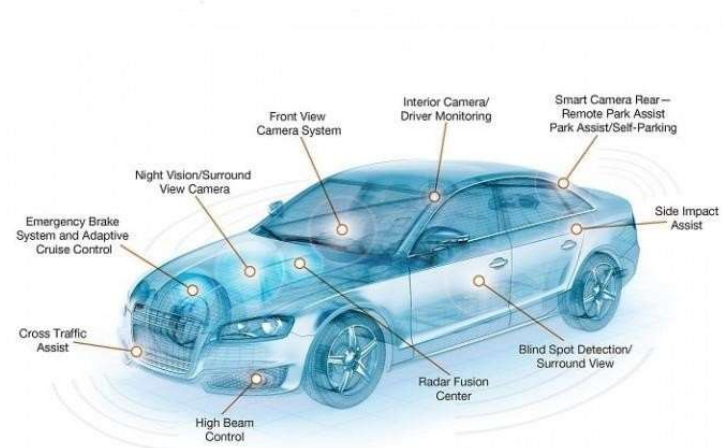
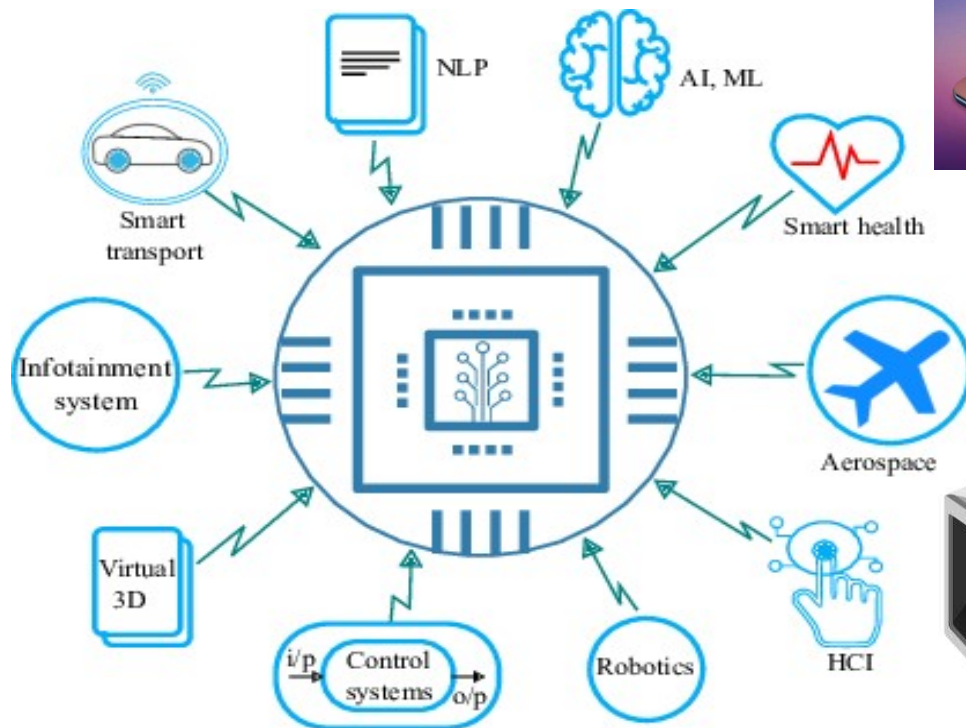
Compiled with reference to other presentations

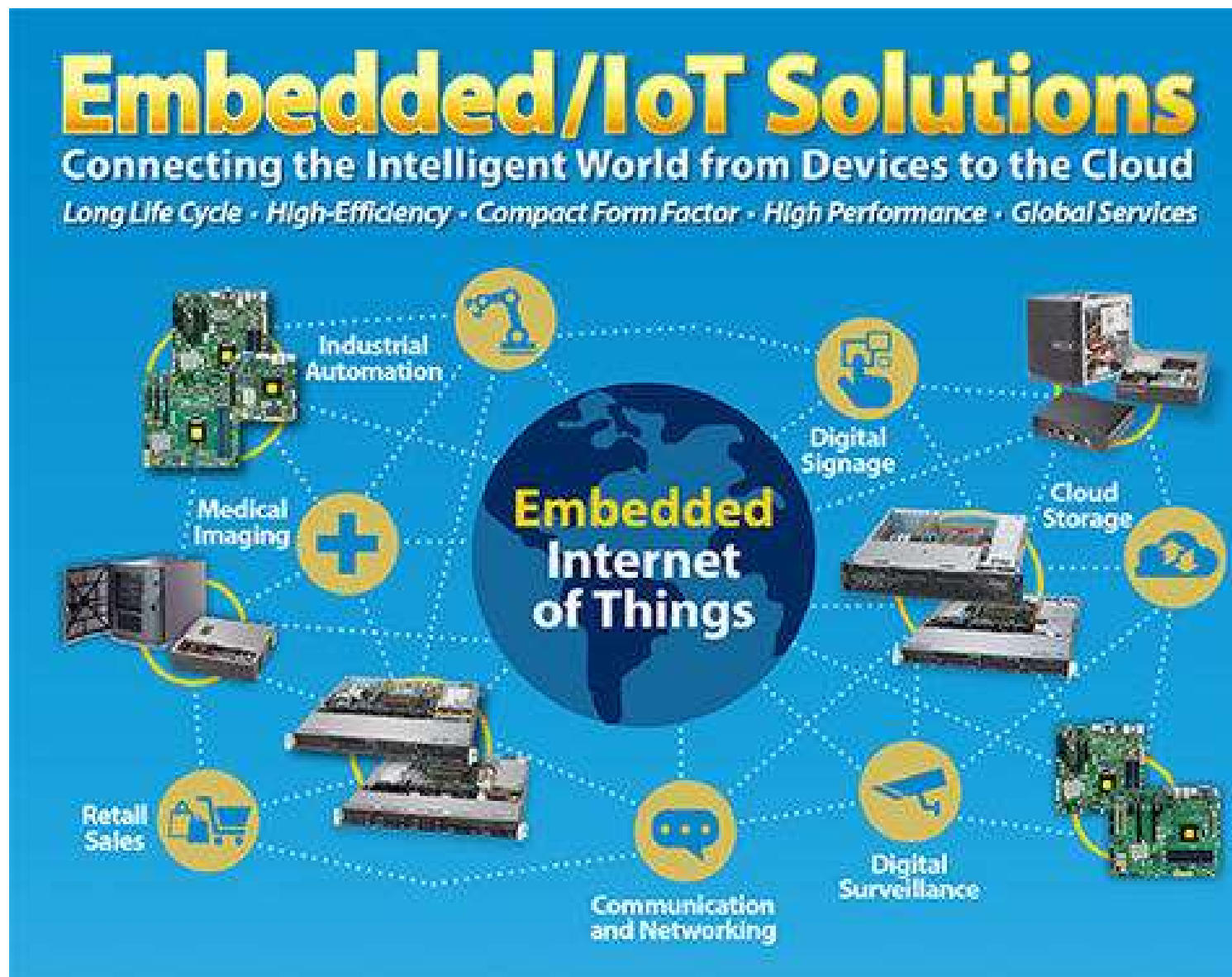
- Concepts
- Applications
- Physical components
- System software
- Design/implementation flow
- Software development



Concept

- Embedded System is **a special-purpose computer system** designed to perform one or a few dedicated functions - *Wikipedia*
- Embedded System market is super competitive





Remarkable features

- Part of larger system
 - *not* a “computer with keyboard, display, etc.”
- HW & SW application-specific – not G.P.
 - application is known a-priori
 - definition and development concurrent
- Interact (sense, manipulate, communicate) with the external world
- Never terminate (ideally)
- Operation is *time constrained*: latency, throughput
- Other constraints: power, size, weight, heat, reliability
- Increasingly high-performance (DSP) & networked



Why are they thriving?

- Market demand: at home, at work, on road, shopping, entertainment, etc
- Hardware maturity: high computing power, mass production, fast-moving VLSI, *[future]* quantum computers
- Prevailing interconnected world – IoT
- Advances in related industries: robotics, digital transformation, sensing, automotive, biomedical engineering, etc

- Even though embedded systems cover a wide range of special-purpose systems, there are common characteristics
 - **Low cost**
 - Should be cheap to be competitive
 - Memory is typically very small compared to a general purpose computer system
 - Lightweight processors are used in embedded systems
 - **Low power**
 - Should consume low power especially in case of portable devices
 - Low-power processors are used in embedded systems



– High performance

- Should meet the computing requirements of applications
 - Users want to watch video on portable devices
 - » Audio should be in sync with video
 - Gaming gadgets like playstation should provide high performance

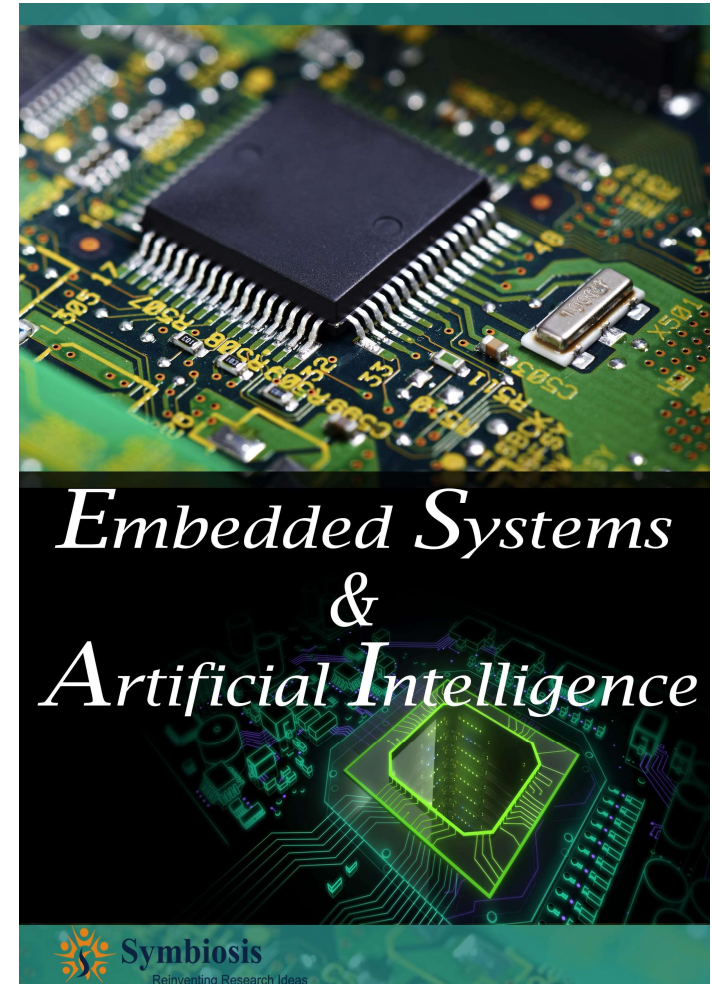


– Real-time property

- Job should be done within a time limit
 - Aerospace applications, Car control systems, Medical gadgets are critical in terms of time constraint – Otherwise, it could lead to catastrophe such as loss of life
- Will talk more about this



- It is challenging to satisfy the characteristics
 - You may not be able to achieve high performance while utilizing cheap components and maintaining low power consumption
 - So, you got to do your best in a given circumstance to be competitive in the market



- Identical to the general computer systems



Application Software

VxWorks



ANDROID

OS / Device Drivers

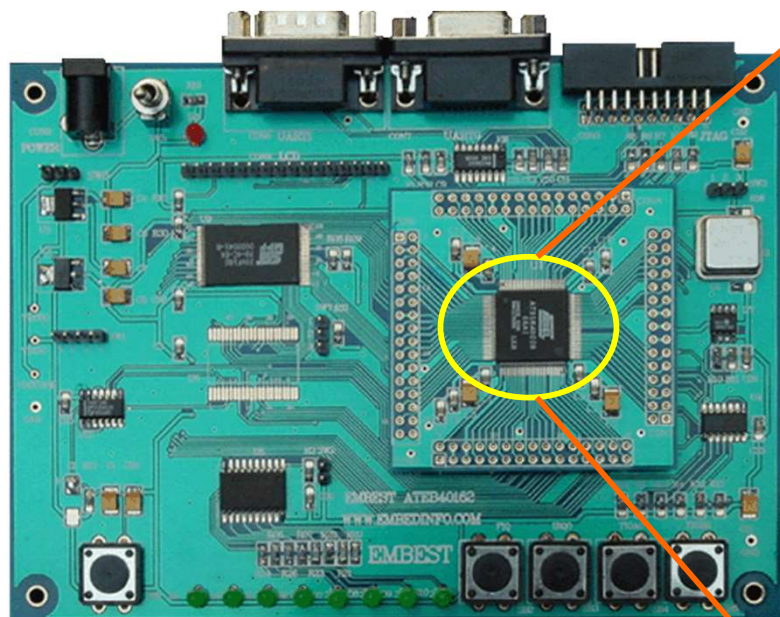
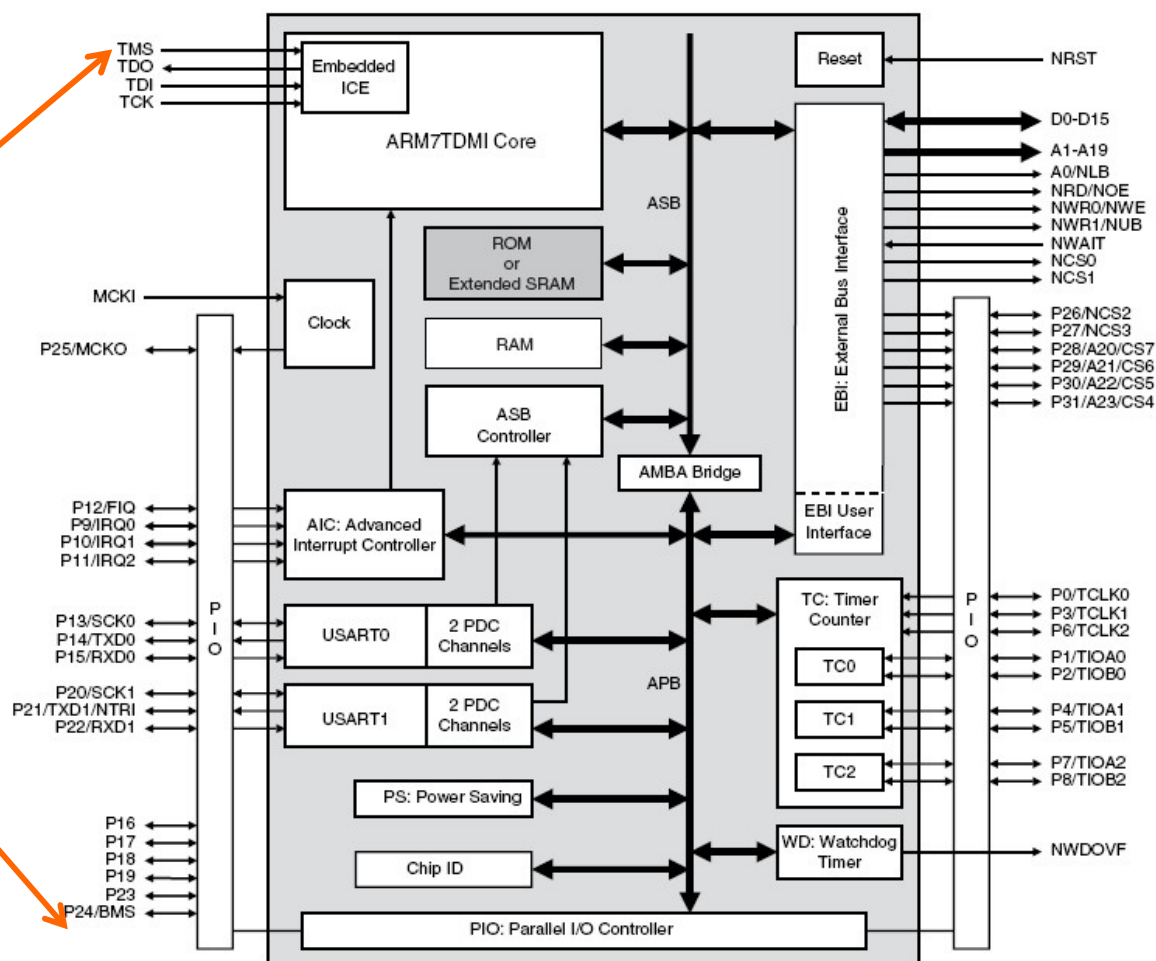


Hardware

Components of Embedded Systems

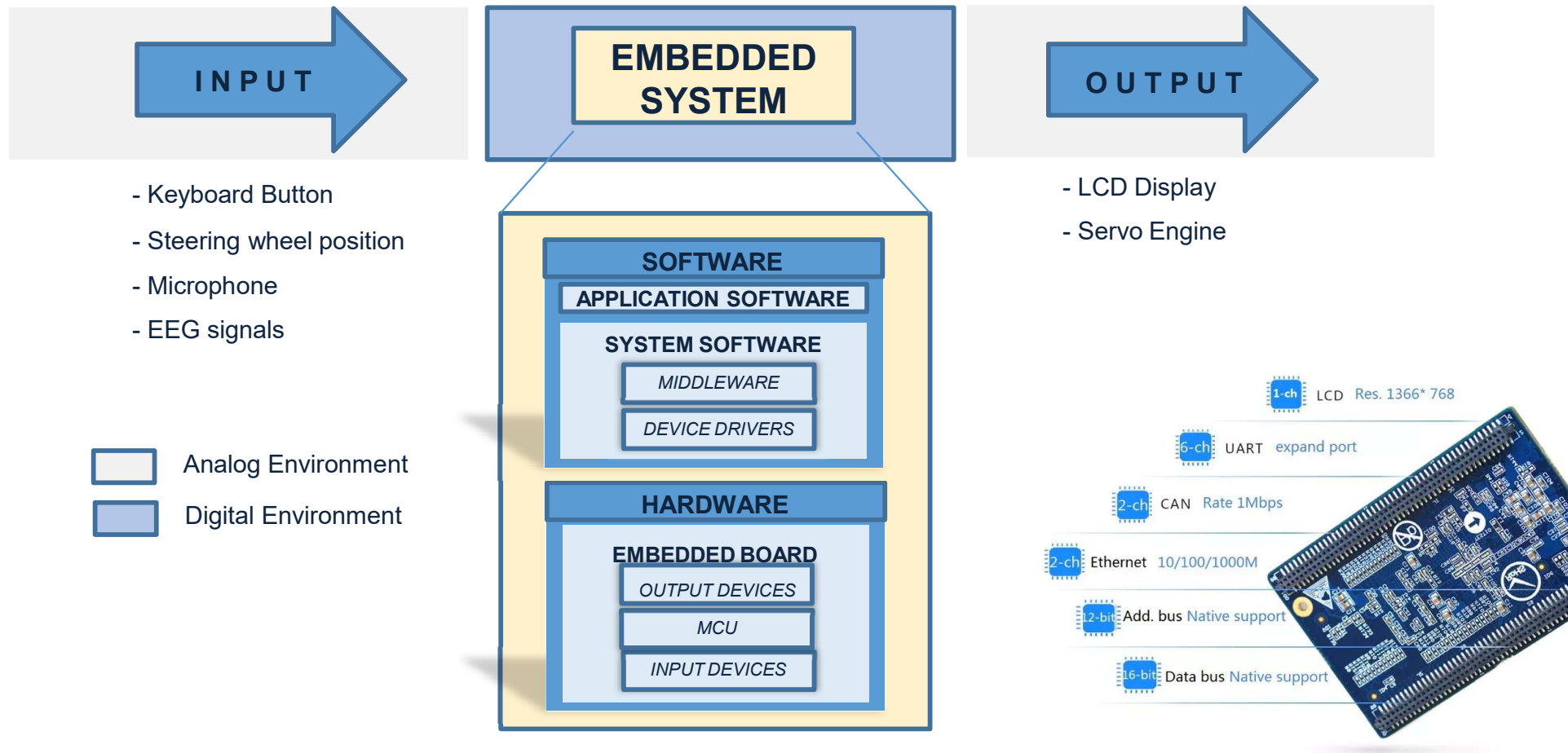
• Hardware

- It is mainly composed of processor (1 or more), memory, I/O devices including network devices, timers, sensors etc.

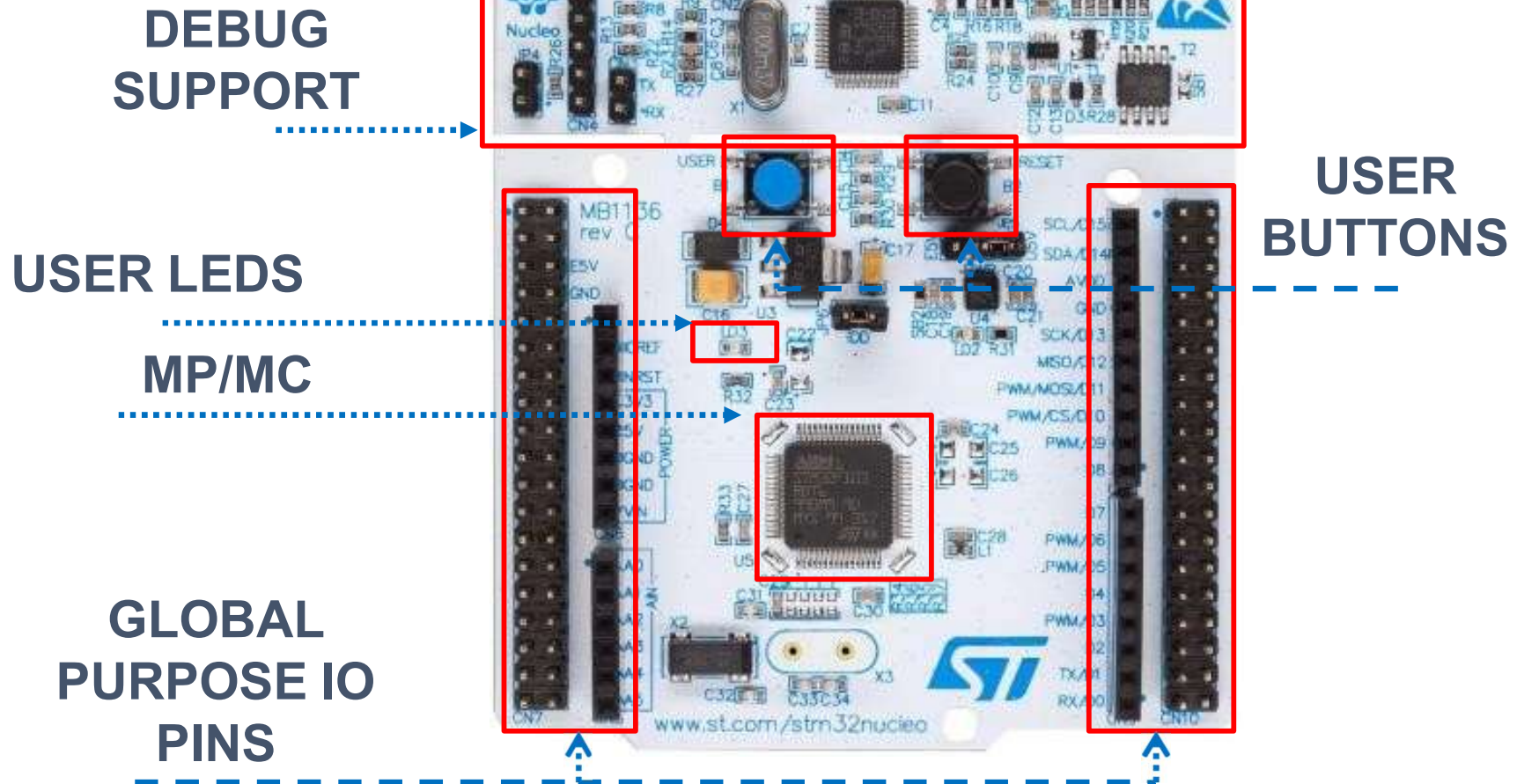


Hardware: system view

Embedded system is a special-purpose computing device designed to perform dedicated function



Typical layout



- 2. Introduction to Embedded Systems - Embedded System Development Components | Coursera

<https://www.coursera.org/lecture/introduction-embedded-systems/2-introduction-to-embedded-systems-5z2lZ>

- **Software - System software**

- Operating systems



iOS



- Many times, a multitasking (multithreaded) OS is required, as embedded applications become complicated
 - Networking, GUI, Audio, Video
 - CPU is context-switched to process multiple jobs
- Operating system footprint should be small enough to fit into memory of an embedded system
 - In the past and even now, real-time operating systems (RTOS) such as VxWorks and uC/OS-II have been used because they are light-weighted in terms of memory requirement
 - Nowadays, heavy-weighted OSs such as iOS, Android, Windows Mobile, and embedded Linux (uClinux) are used, as embedded processors support computing power and advanced capabilities such as MMU (Memory Management Unit)

- Device drivers for I/O devices

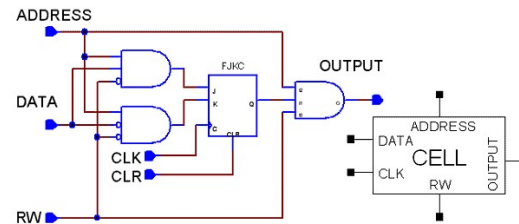
- **Software (cont.) - Application software**
 - Run on top of operating system
 - Execute tasks that users wish to perform
 - Web surfing, Social Network Service, Audio, Video playback



Embedded System Design Flow



**Planning & Architect
(modeling &
simulation)**



ASIC/FPGA/SoC design

**Hardware Design
with CAD tools**



ASIC/SoC chip



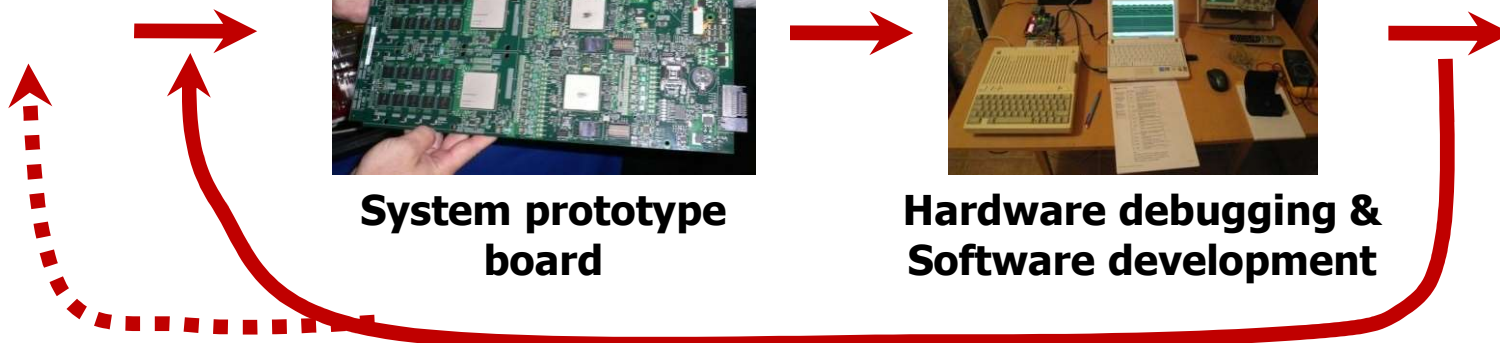
**System prototype
board**



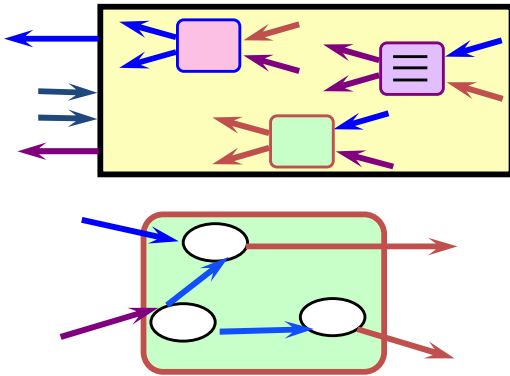
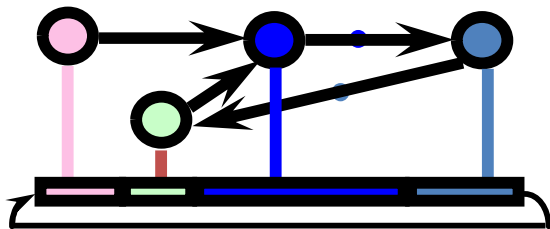
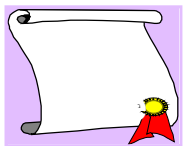
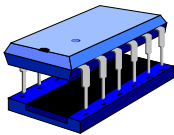
**Hardware debugging &
Software development**



Final product



Environ-
-ment



Modeling

the system, and algorithms involved;

Refining (or “partitioning”)

the function into smaller, interacting pieces

Test Bench Design

Abstractions from the Design, Communication, Storage, and Computation resources

HW-SW partitioning: Allocation

(1) HW

(2) SW

Determine power and performance bounds

Scheduling

When are functions executed

Resource Arbitration

Mapping (Implementing)

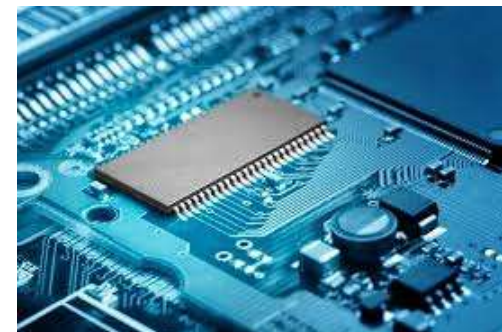
(1) software, (2) Hardware,

(3) Coherence/Communication

Testing/Validation

Insitu with Design

Still interesting ?



Many Implementation Choices

- Microprocessors
- Domain-specific processors
 - DSP
 - Network processors
 - Microcontrollers
- ASIPs (application-specific instruction set processors)
- Reconfigurable SoC
- FPGA
- ASIC
- Full-Custom (COTS)

Speed

Power

Cost



High

Low

Volume

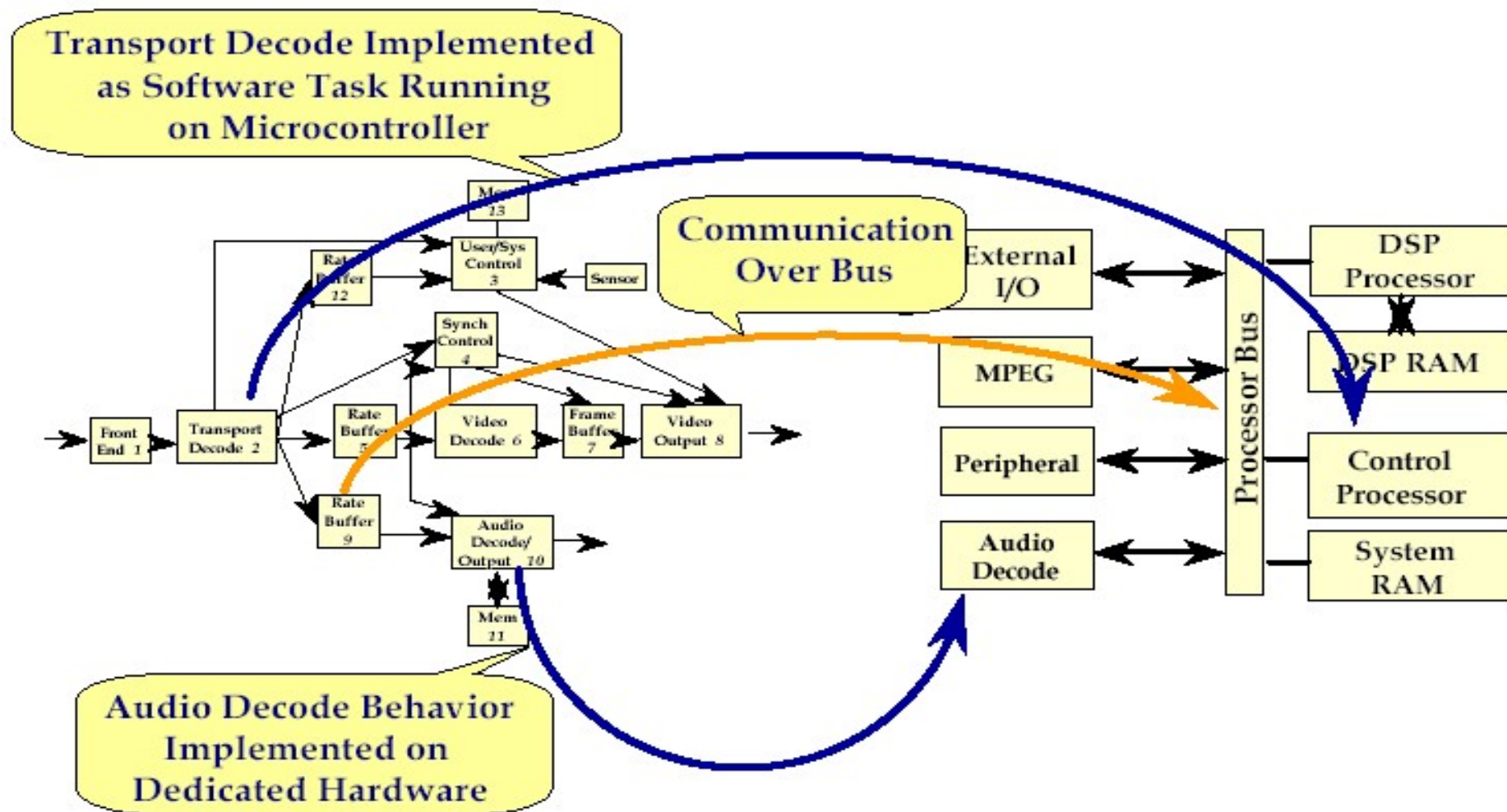
- Good:
 - High performance custom peripherals
 - Multiple heterogenous Cores
 - Integrated A/D, D/A, timers ...
 - Low Cost, High performance on-chip communication
 - Low Part Cost in Volume
- Bad:
 - Very expensive (\$5-\$75M/design)
 - Very High Risk (Several total failure points)
 - Potentially impossible to Program even if working!

Hardware vs. Software Modules

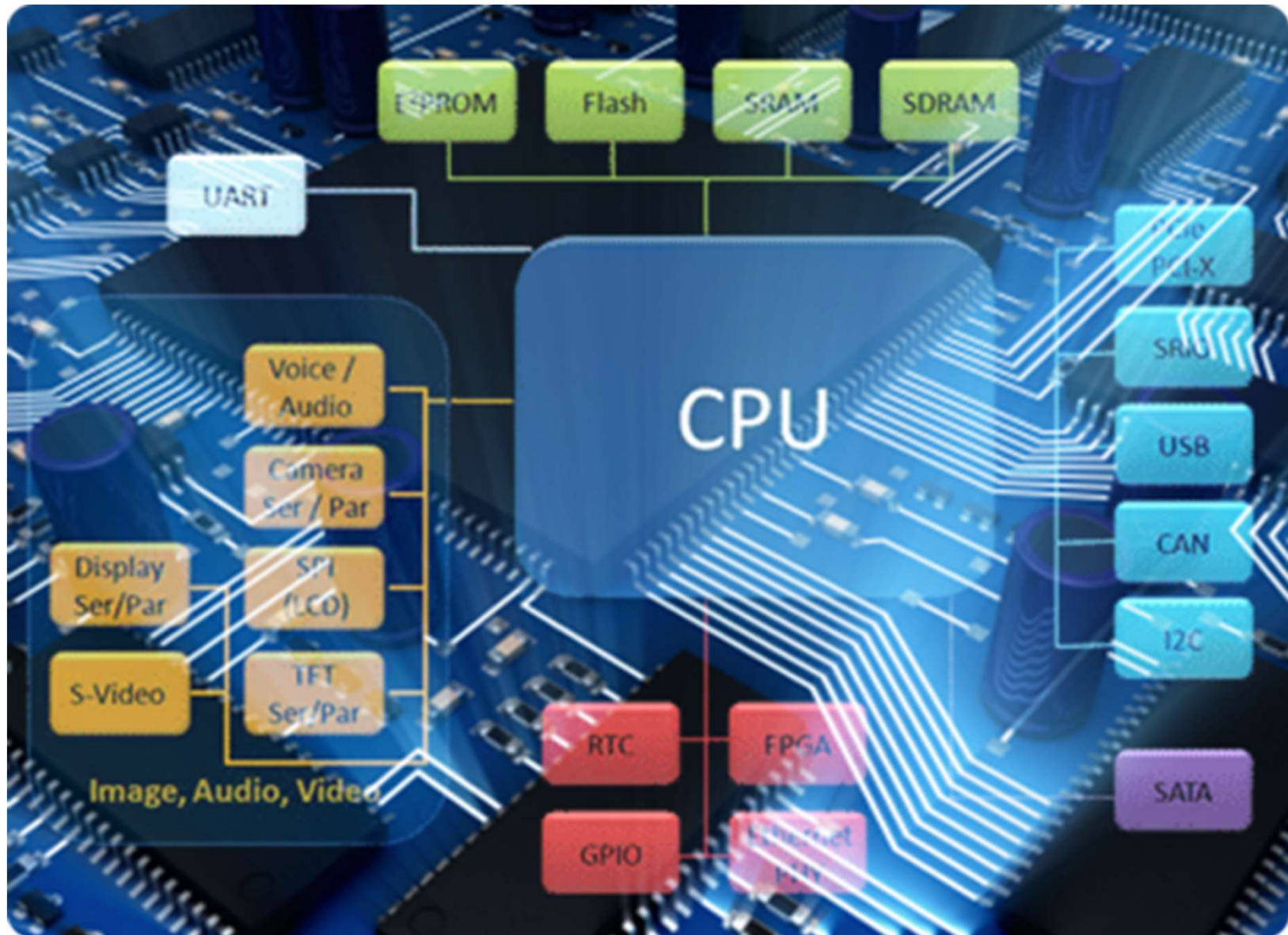
- Hardware = functionality implemented in custom architecture
- Software = functionality implemented stored program
- Key differences:
 - Configurability, Adaptability
 - Process Time Multiplexing
 - software modules time multiplexed on a processor
 - hardware modules often mapped to dedicated hardware
 - Concurrency
 - processors have serial “thread of control”
 - dedicated hardware has concurrent activity
- [Lecture 1.2: More on Embedded Systems - Embedded Systems | Coursera](https://www.coursera.org/lecture/iot/lecture-1-2-more-on-embedded-systems-RnvF2)
<https://www.coursera.org/lecture/iot/lecture-1-2-more-on-embedded-systems-RnvF2>

- A significant part of the system design problem is deciding which parts are software, and which are hardware
- Issues:
 - Cost of development
 - System performance
 - Upgrade potential
 - Sales/Implementation Volume
 - Availability of IP
 - Development Time-Line

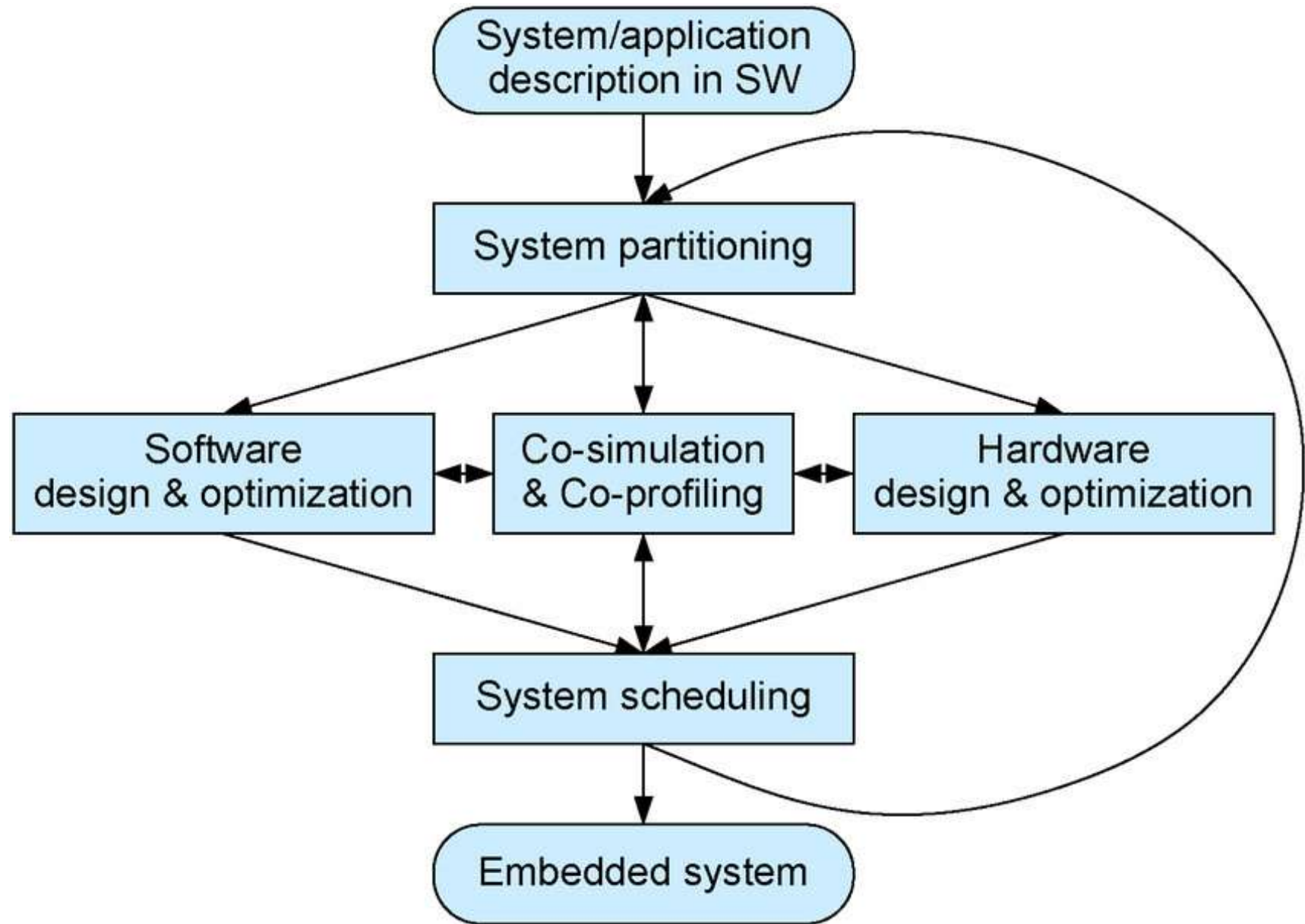
Map Behavior to Architecture

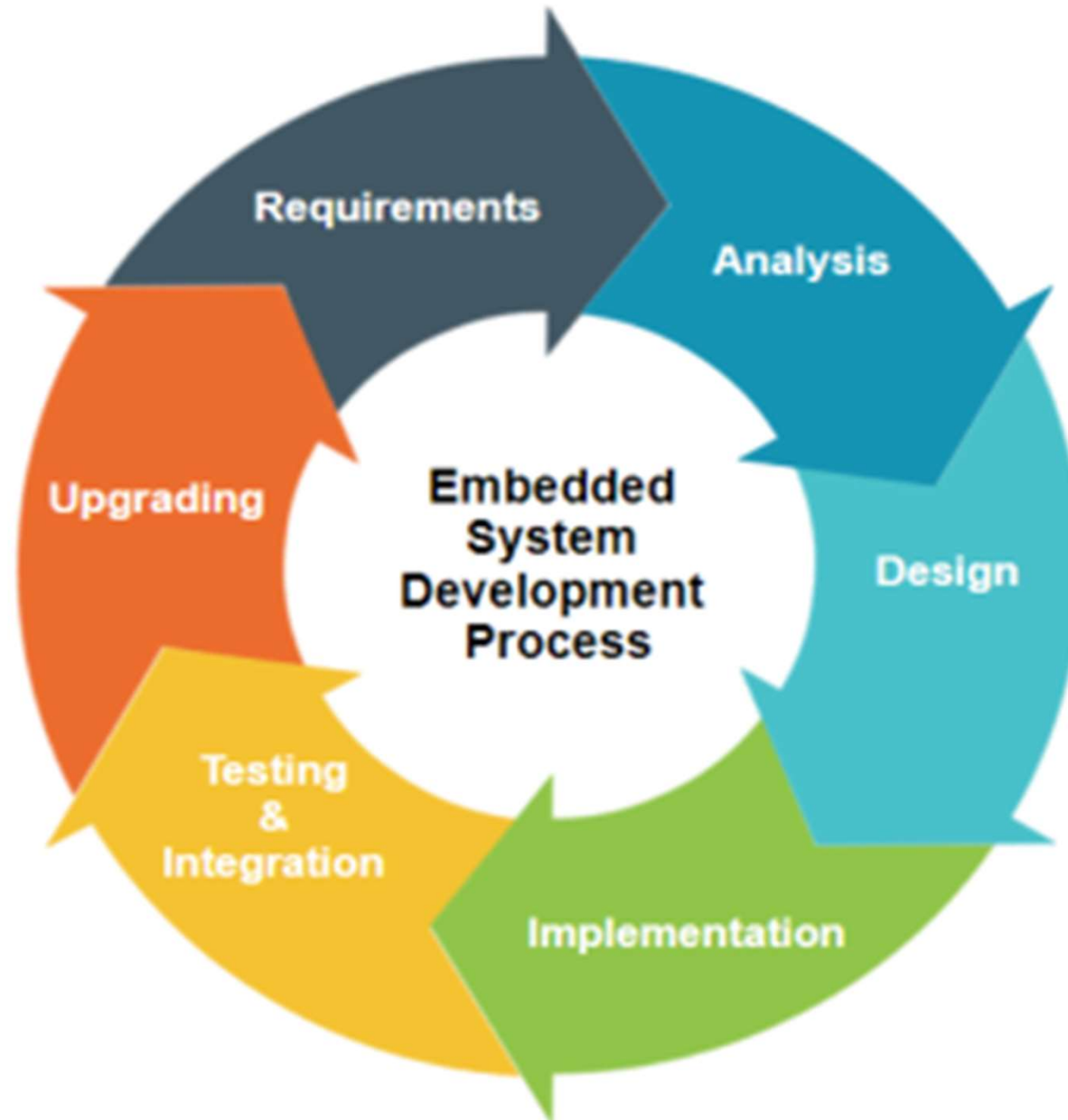


A more specific image

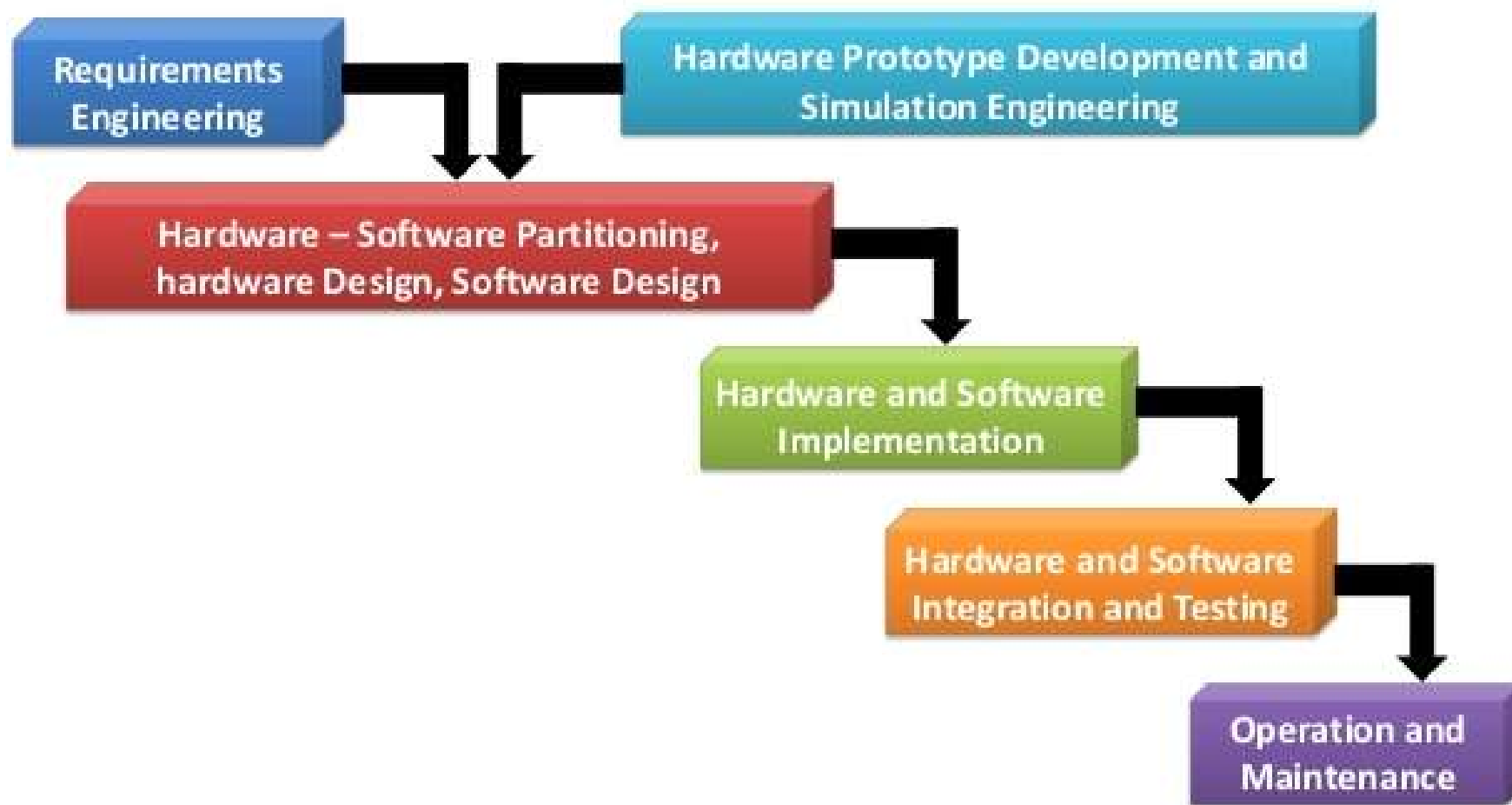


Typical design flow

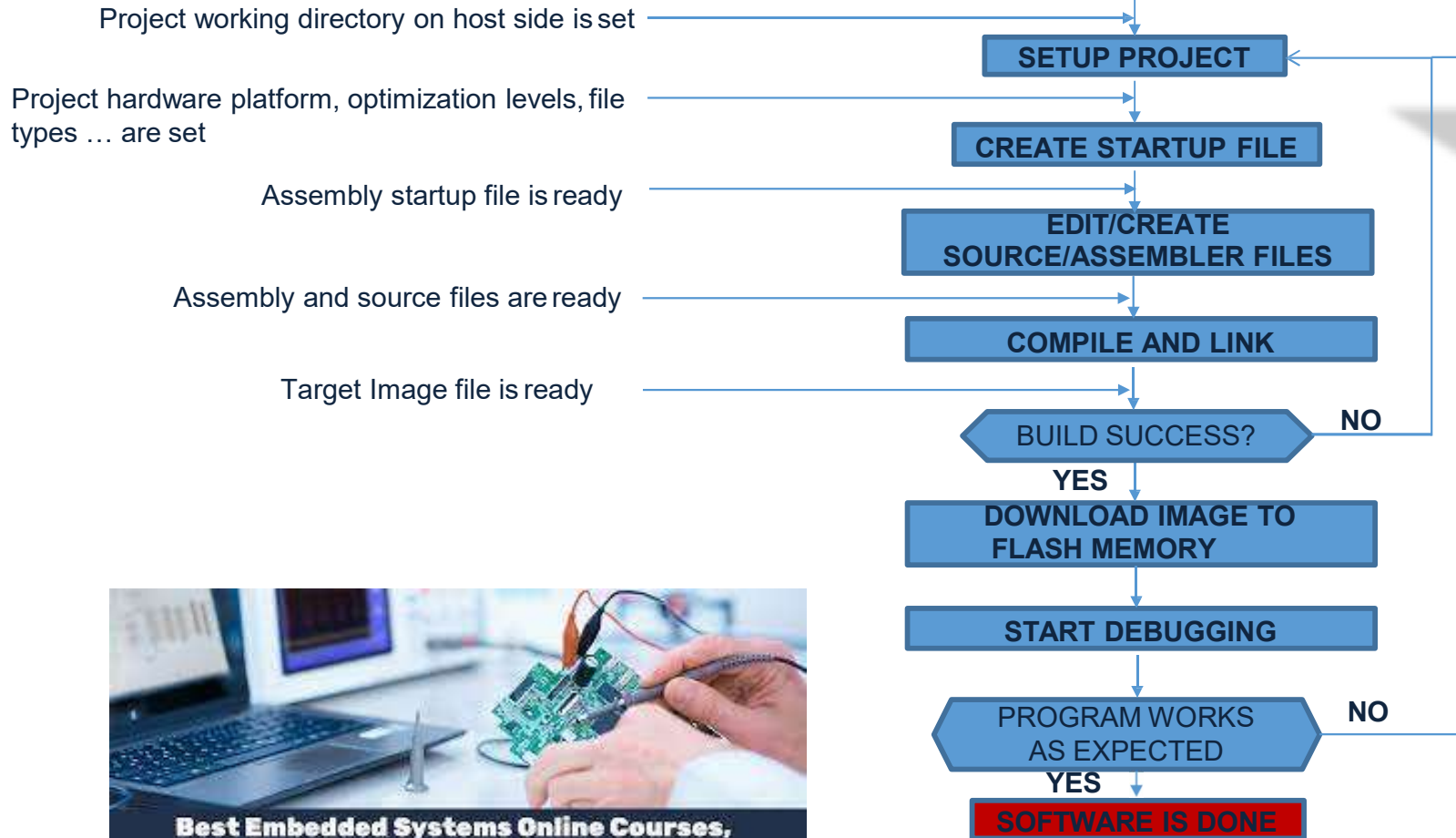
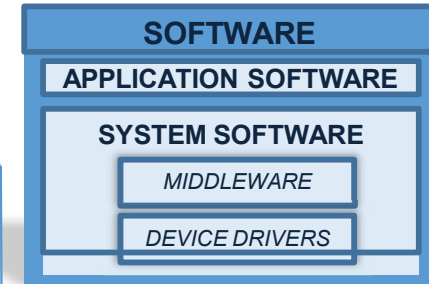




Waterfall model for Process of Embedded System Development

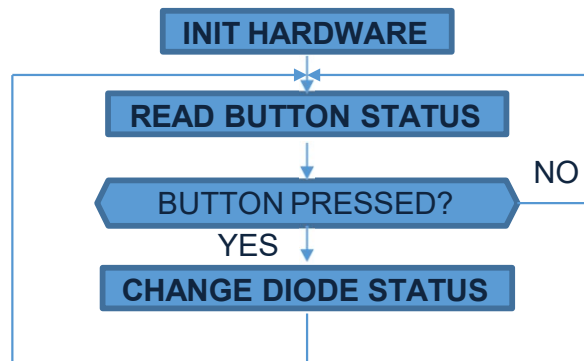


Software Project Development Process Flow



„main“ source code development for our project example

1. Develop Software model



```

int main(void)
{
    ButtonStatus ButtonStatus =
        RELEASED;
    InitHardware();
    while (1)
    {
        ButtonStatus =
            ReadButtonStatus();
        if(ButtonStatus == PRESSED){
            ChangeDiodeStatus();
        }
    }
}
    
```

2. Map model to the code

- **INIT HARDWARE** - hardware initialization (clocks and peripherals initialization)
- **READ BUTTON STATUS** - read pin status from GPIOC_IDR register and return corresponding result
- **CHANGE DIODE STATUS** - toggle pin value in GPIOA_ODR

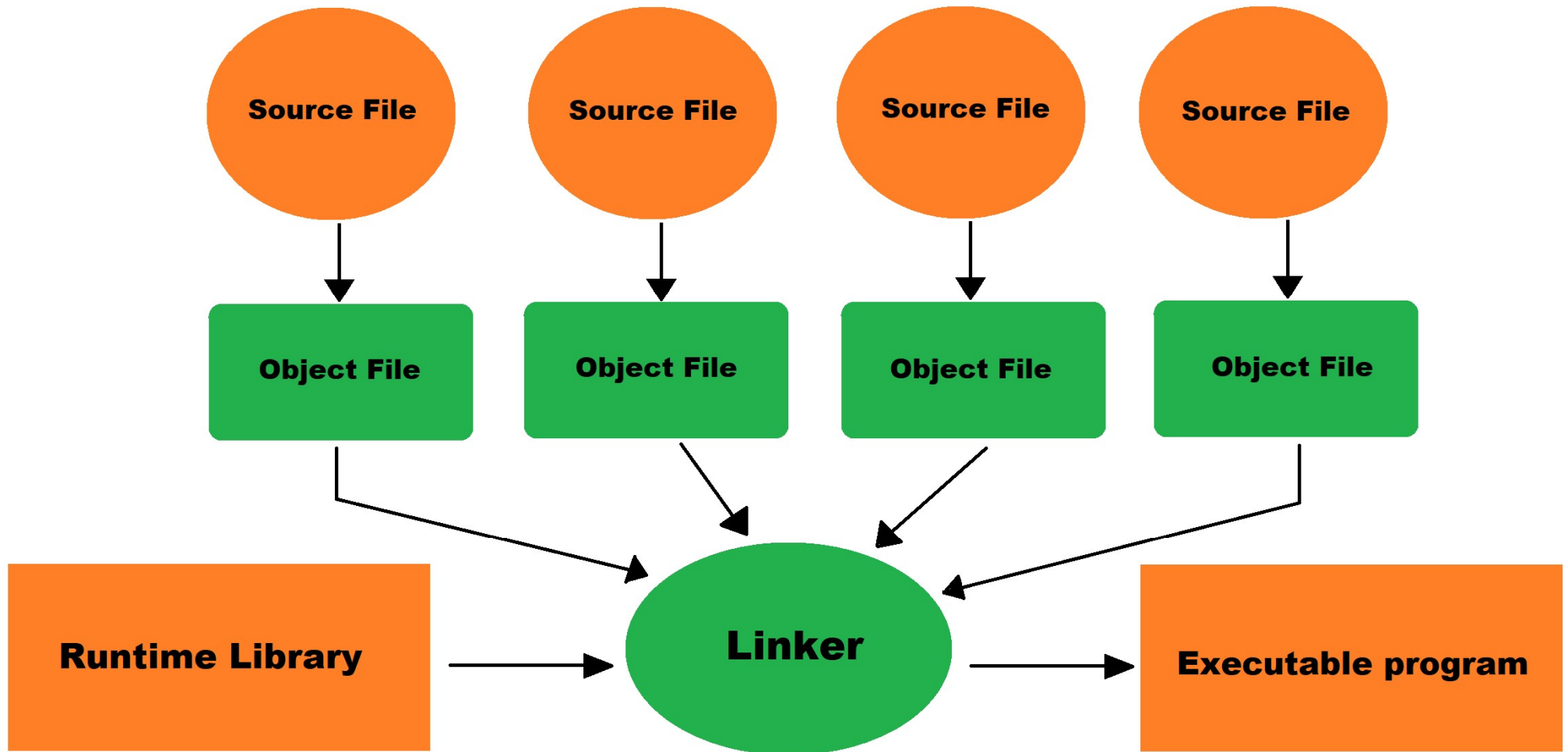
3. Write code on host side

```

typedef enum{
    PRESSED,
    RELEASED
}ButtonStatus;
ButtonStatus ReadButtonStatus(){
    if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13) == 0) return
        PRESSED;
    return RELEASED;
}
void ChangeDiodeStatus(){
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
}
    
```

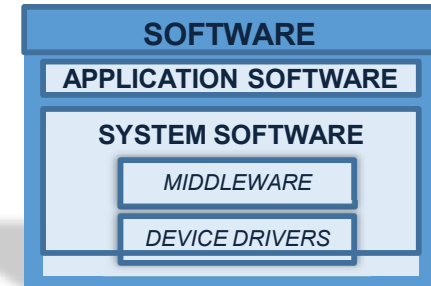
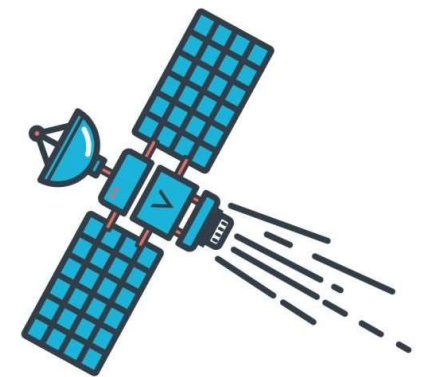
What is Embedded Software?

➡ **Embedded Software is the software that controls an embedded system.**



Build process

- From user code (assembly and C code) create executable code which will be loaded to the target memory
- Compiler
 - Translate programs written in some human-readable language into the equivalent set of opcodes for particular processor
 - Cross-compiler
 - ❑ Code is written on a different platform and then compiled for embedded platform
 - Compilation result is object file which corresponds to the dedicated source file
- Assembler
 - “Compiler for assembly language files”
- Linker
 - Combine object files and locate them in memory
- Result



TARGET IMAGE FILE

Embedded Systems Software Development Tools



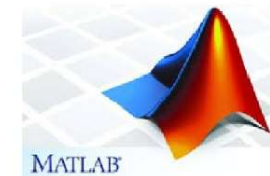
MPLAB



**Arduino
Software**



PSpice



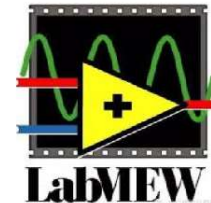
MATLAB



Proteus



**Visual
Studio**



LabVIEW

ARMKEIL
Microcontroller Tools

Keil

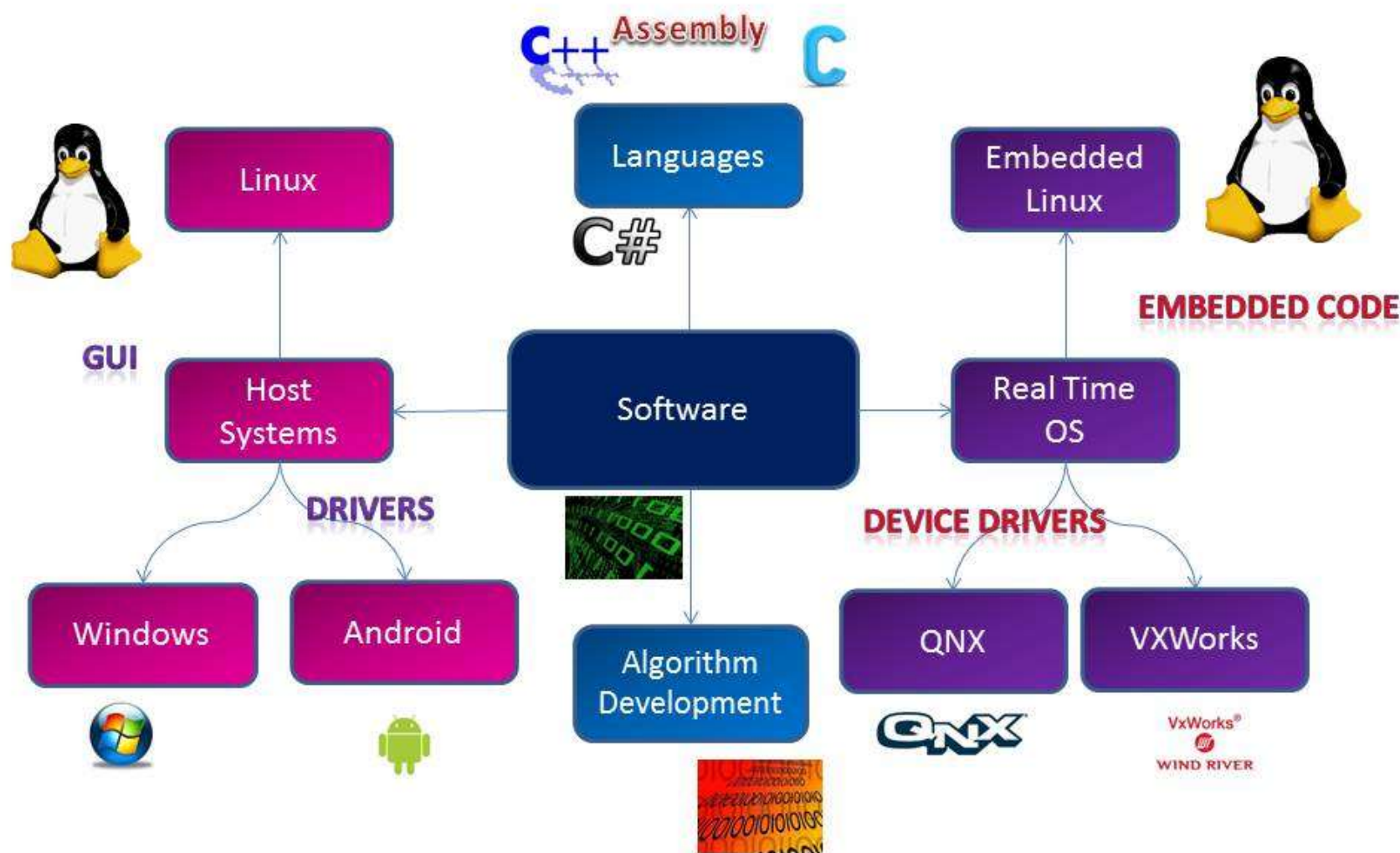
www.TheEngineeringProjects.com

Enterprise products

Nios II IDE 	MCUXpresso 	Code Warrior 	TrueStudio 	Sloeber (for Arduino) 	Kalray 	iDev 
VX Software 	Artik IDE 	e2 studio 	Momentics 	Code Composer 	Sourcery CodeBench 	Simplicity Studio 
DAVE 	Xtensa Xplorer 	DS-5 	CrossCore (CCES) 	eGui 	Ascet Developer 	Cevelop 
XSDK 	Luminosity 	SoftConsole 	Snapdragon Debugger 	Wind River Workbench 	System Workbench 	COSIDE® 

Eclipse CDT (www.eclipse.org/cdt/) based IDEs, tool suites and plug-ins, 2017

Programming point of view



Eclipse organization showing top-level projects

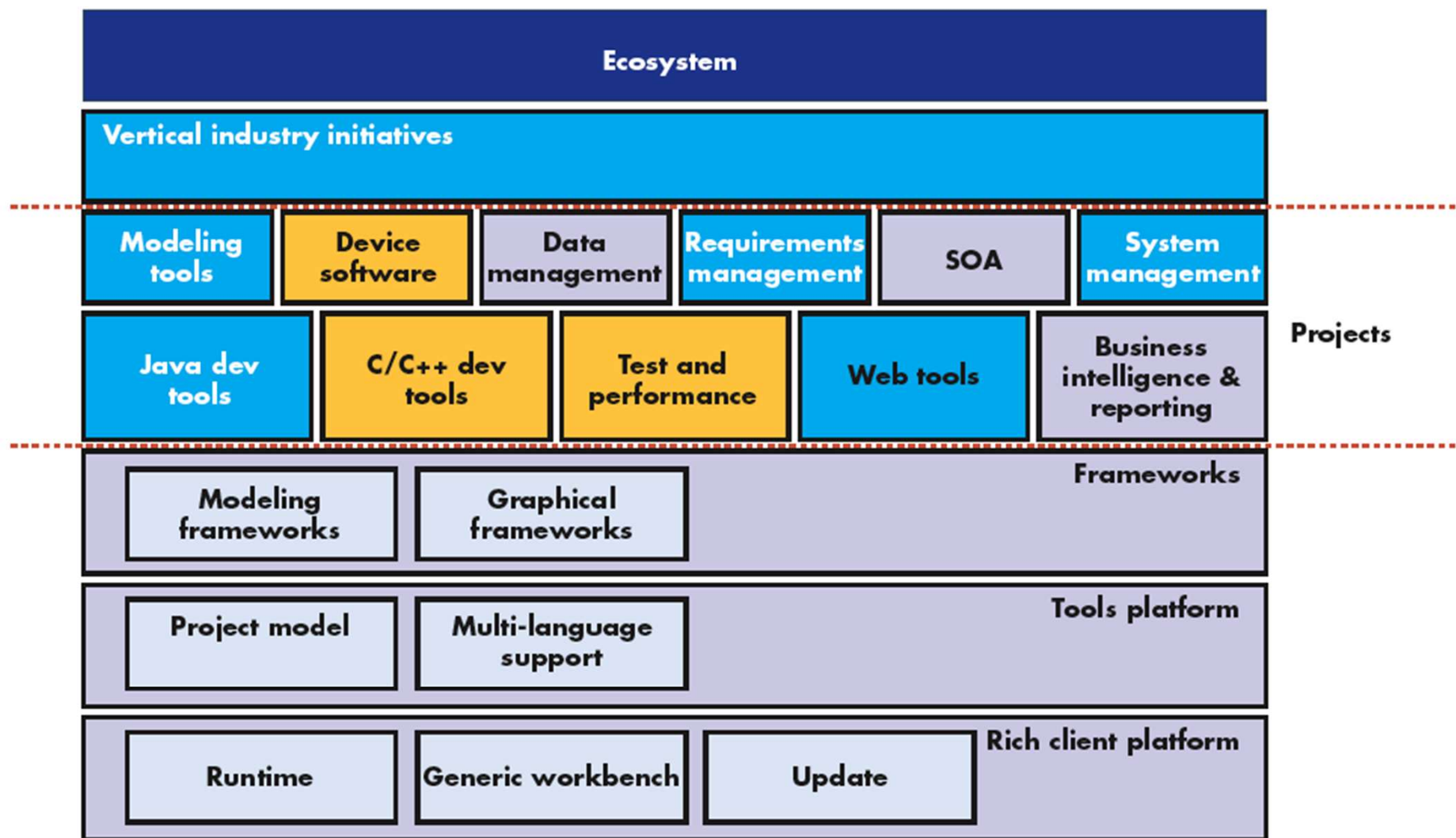
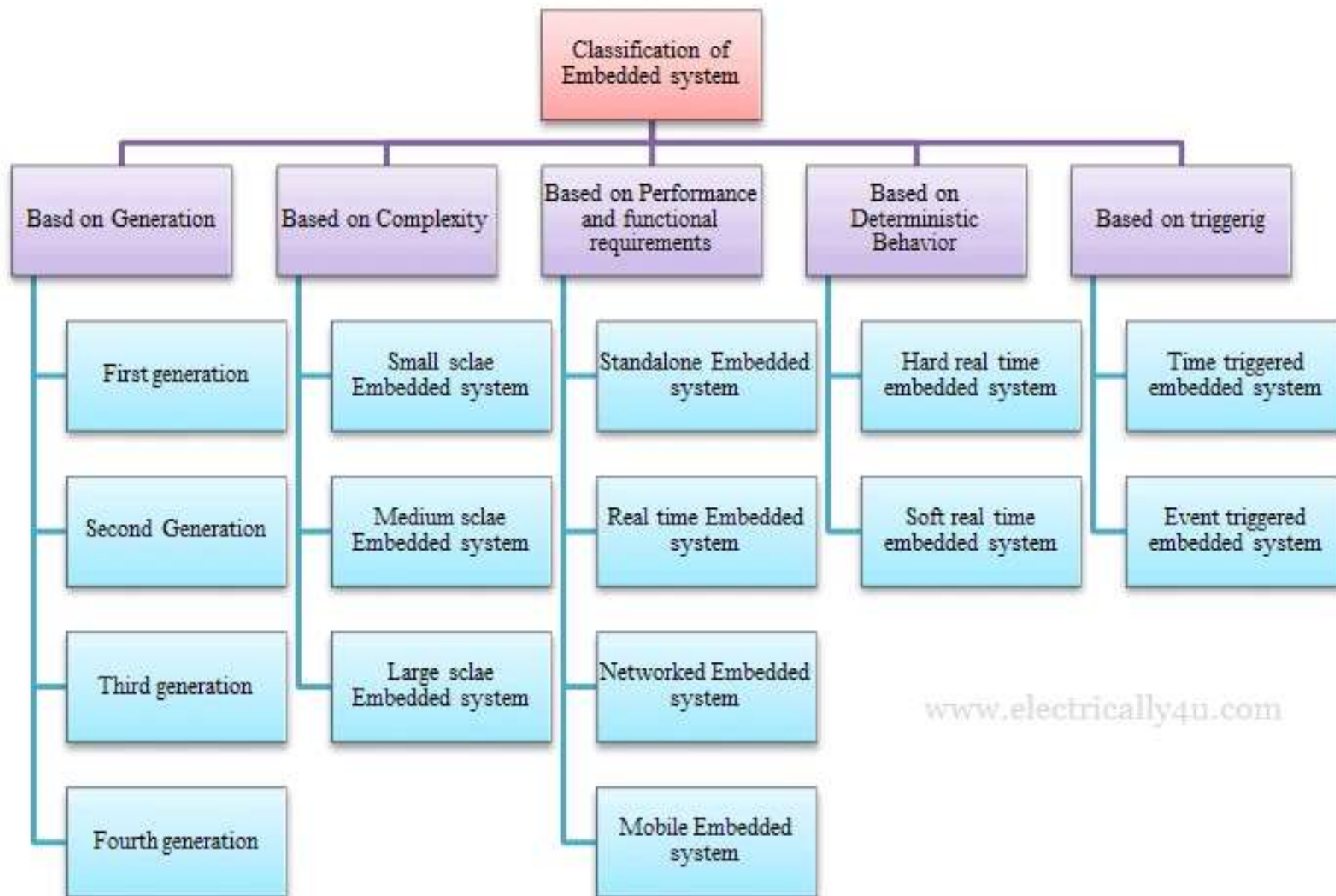


Figure 2

Grandview classification



- ES-Unit1-L4-classifications of embedded systems – YouTube

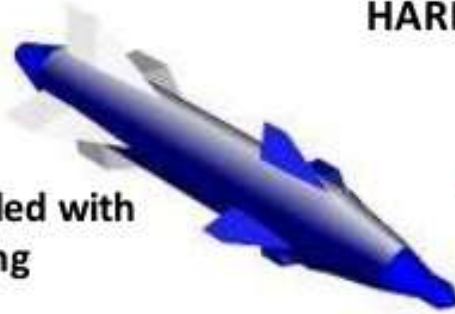
<https://www.youtube.com/watch?v=dd7uxDgF-k0>

Real Time Embedded Systems

A system in which work has to be done in a specific time period.

HARD REAL TIME

Missile
embedded with
a tracking
system



SOFT REAL TIME



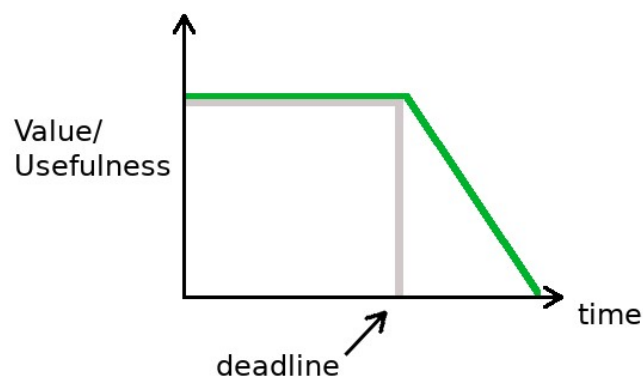
Aircraft



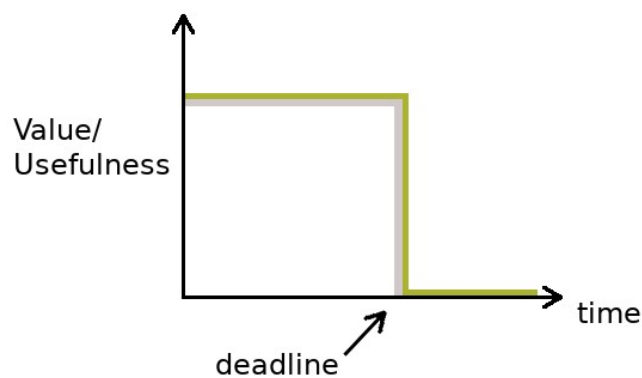
DVD PLAYER

Hard- vs. Soft

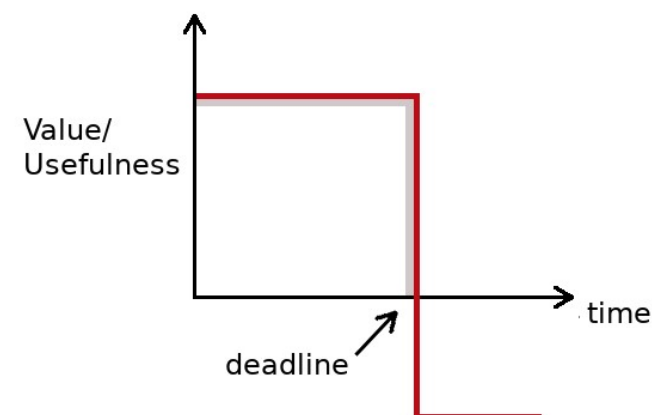
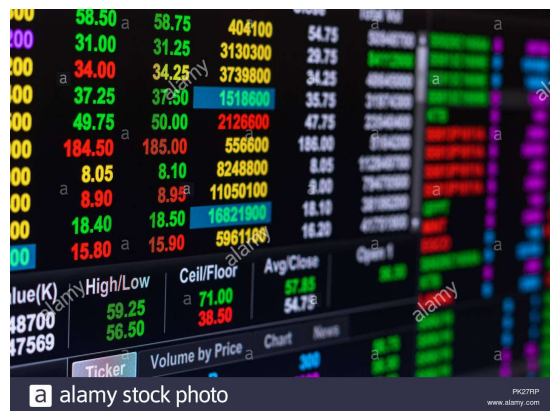
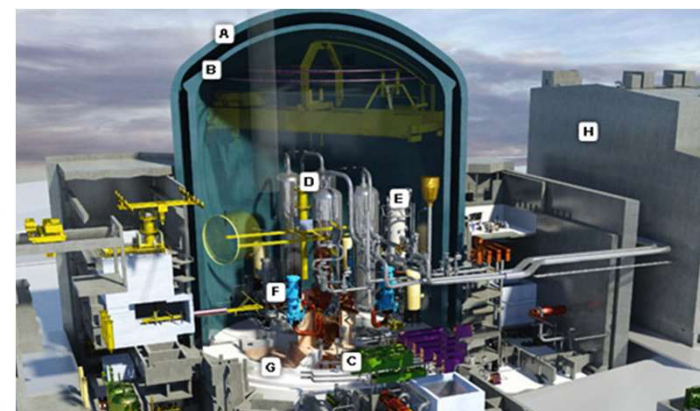
soft real-time



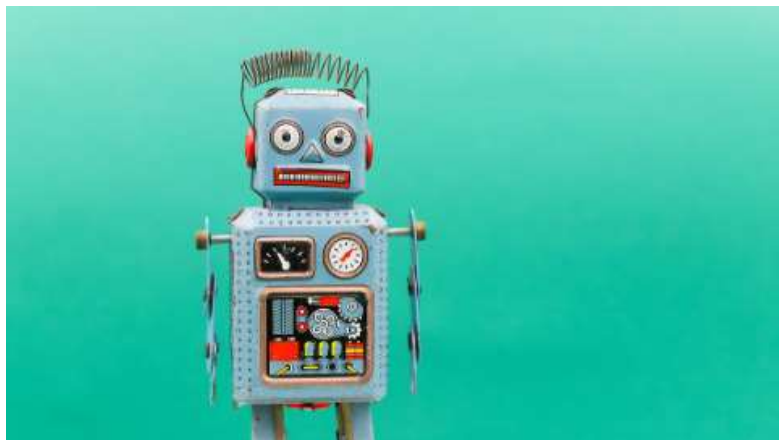
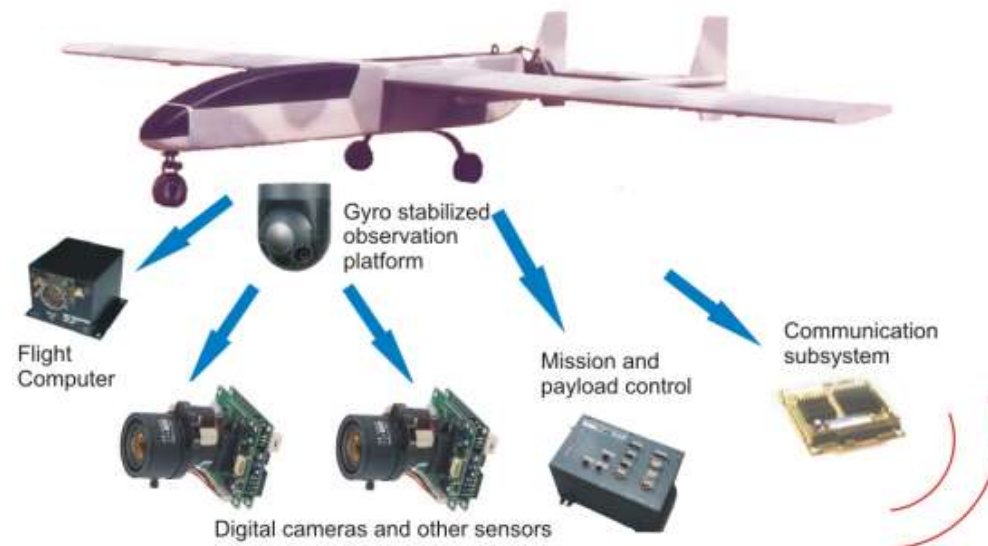
firm real-time



hard real-time



Static vs. Dynamic



■ Static

- Scheduling decisions are based on fixed parameters (e.g. priority), assigned to tasks before their activation

■ Dynamic

- Scheduling decisions are based on dynamic parameters (e.g. priority) that may change during task execution

- <https://youtu.be/BxYwjdrdnQg?t=225>