Embedded Systems

(Embedded Real-Time Systems)

Teaching staff:

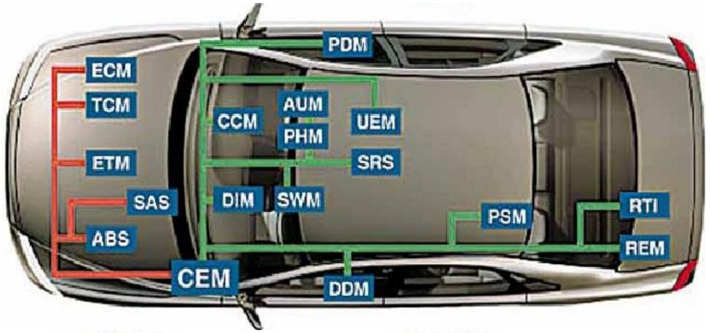
Luís Almeida (lda@fe.up.pt)

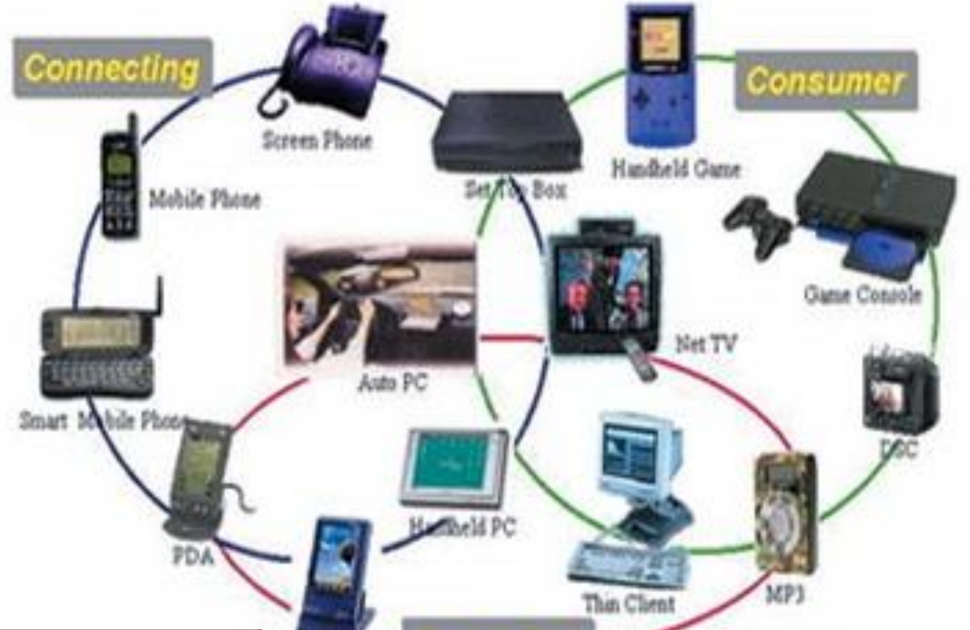
Mário Sousa (msousa@fe.up.pt)

Background info

Embedded System

- **Computing** system
- **Immersed** in a system/device that has a **specific purpose**
- Connected to that system/device through **specific input/output**
- Typically **unfit** to carry out **other functionality**
- Typically subject to **diverse constraints**:
    - dimension, cost, reliability, safety security, **real-time**…

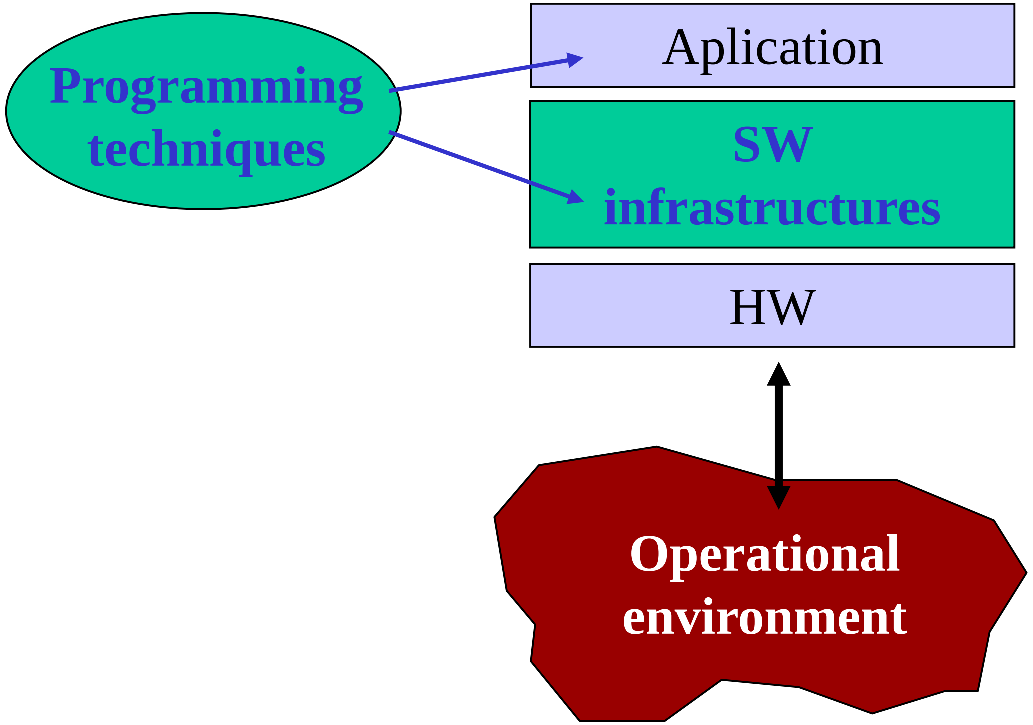




Scope of this course

Main topic:

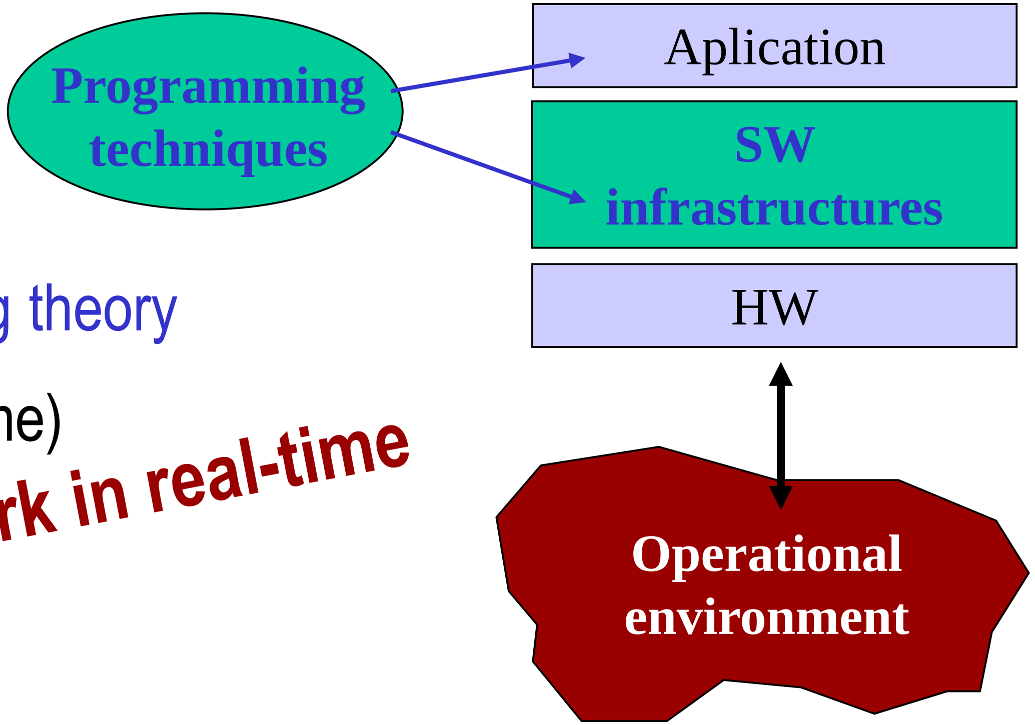
- Embedded systems **programming**
    - **Software** infrastructures and **programming** techniques for embedded systems (with focus on **real-time systems**)



Objectives of this course

Provide education and training in:

- Identifying and characterizing the constraints imposed on an embedded system with focus on the temporal constraints
- Deciding the most suitable approach to track the environment system state
- Defining and managing concurrent activities and analyze their behavior with (real-time) scheduling theory
- Choosing, using and building embedded (real-time) operating systems



Designing systems that work in real-time

FAQ – on real-time aspects

Working in real-time... Isn’t it enough to use a fast processor?

- If the program has a **trivial control structure**, e.g., a single loop, probably yes!
- If the program includes **multiple concurrent threads of execution**, processing **speed** isn’t enough. Some of the threads can **interfere** with others causing **delays** that might jeopardize real-time operation!

FAQ – on real-time aspects

If not just a fast processor... then what is necessary?

- Proper **scheduling** ! i.e., correct **execution order** that may allow each concurrent thread (task) to finish and **generate its outputs in time** to keep up with the pace of the environment.
- There are specific scheduling techniques that allow us to **bound** and **determine a priori** the **maximum delay** that a task can suffer

FAQ – on real-time aspects

Concurrent threads... Then it only applies to *multitasking* OS?

- **Yes**, without concurrent tasks scheduling does not make sense
- **No**, even with single loop programs there may be **hidden concurrent threads**, e.g., asynchronous interrupt service routines!

Attention !

FAQ – on real-time aspects

And why are such delays so important?

What are we talking about?

- The avionics in an airplane? A steer-by-wire system in a car? The trajectory control in a rocket?
  → **delays** imply **actuating late** → potential **instability** and **loss of control**

Potential catastrophy

- An MPEG player? A cellphone? A multimedia games console?
  → **delays** imply **missing frames/calls** → **degradation of quality of service**

Annoying…

Bibliography

Preferential

- G. Buttazzo. ***Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*** *(2nd ed.).* Springer. (scheduling)
- H. Kopetz. ***Design Principles for Distributed Embedded Applications*** *(2nd ed.)* Springer. (temporal constraints, temporal control, dependability)

Complementary

- Jane W.S. Liu. ***Real-Time Systems****.* Prentice Hall.
- Welling, A. and A. Burns. ***Real-Time Systems and Their Programming Languages*** *(3rd ed.).* Int. Computer Science Series, Addison-Wesley
- Rômulo Silva de Oliveira, ***Fundamentos dos Sistemas de Tempo Real*** (in Portuguese), Material.

Course organization

Lectures **–** presenting and discussing concepts and techniques

- Concentrated in the **first half** of the semester
- Keep an eye on the recommend bibliography
- Slides and videos (in portuguese) available on the course wedpage
- Seminars with presentations of selected topics by groups of students ← **for assessment**

Laboratory **–** applying those techniques in concrete use cases

- Concentrated in the **seconf half** of the semester
- Diverse platforms: RaspberryPI (ARM11), ICnova (AVR32), microcontrollers (ATmega…, PIC…)
- Set of guided experiments to provide contact with embedded platforms
- **One project per group** (groups of 3 students)

Assessment

- **Final grade** will be determined by:

Normal period

• Lectures: **50%** (40% written exam, 10% seminars)
• Laboratory: **50%** (25% demo/discussion, 25% project report)

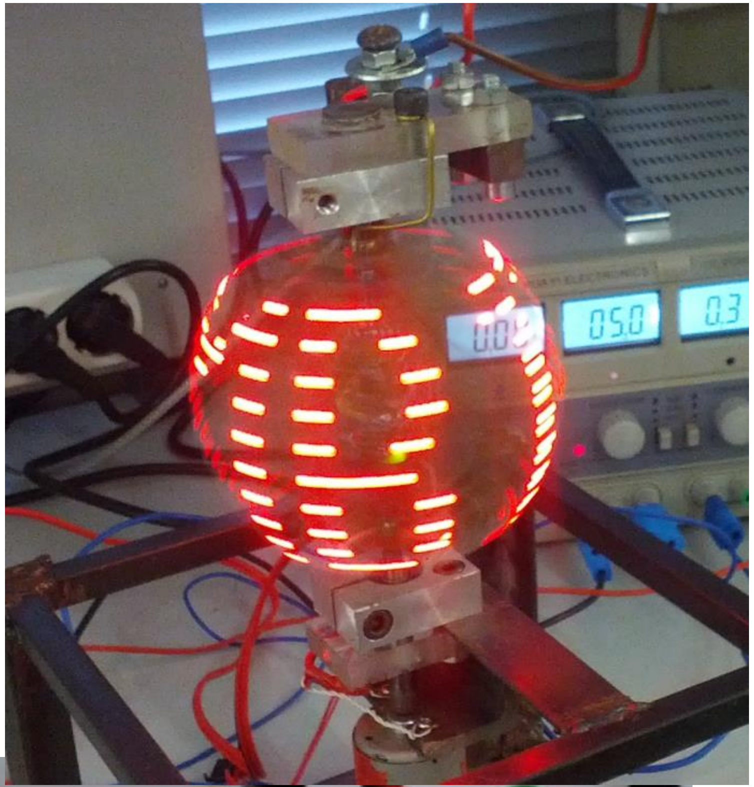
Recourse period:

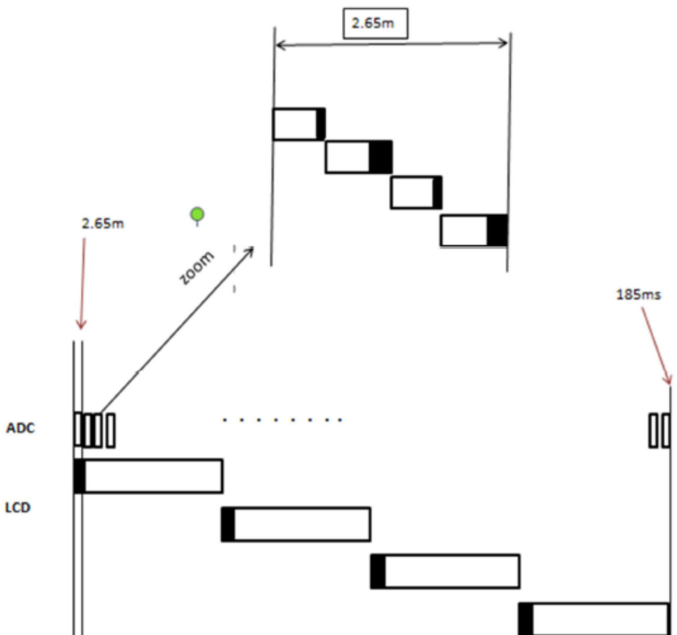
• **written exam**, replaces the normal period exam grade if better

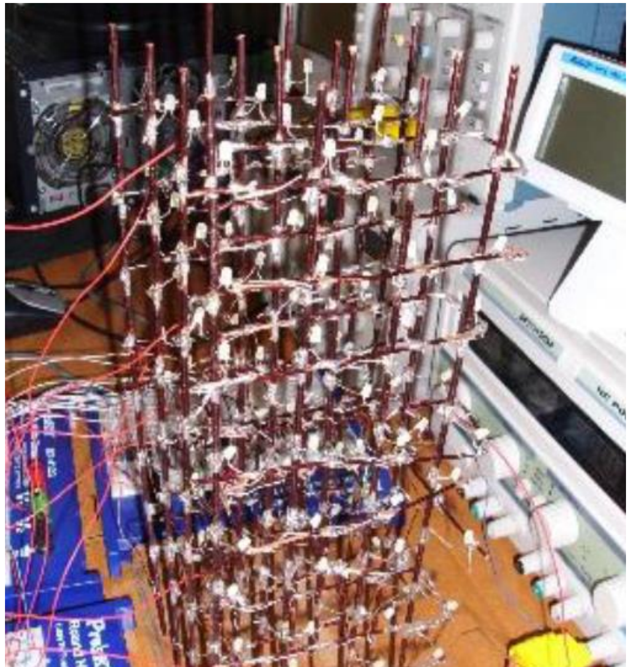
Obs: Minimum grade of 7/20 is required for the **exam**, **exam+seminars** and **project**

Examples of projects

- Controlled switching power supply
- POV – Persistance of Vision devices (3D)
- Preemptive kernel for 8-bit microcontrollers
- Interactive in a tower of LEDs

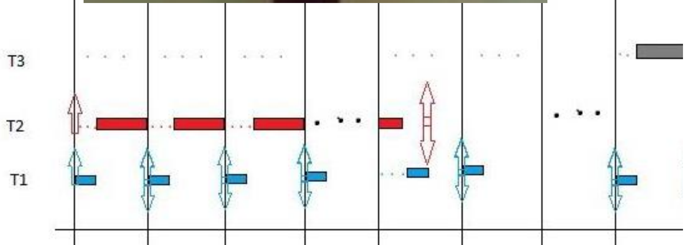


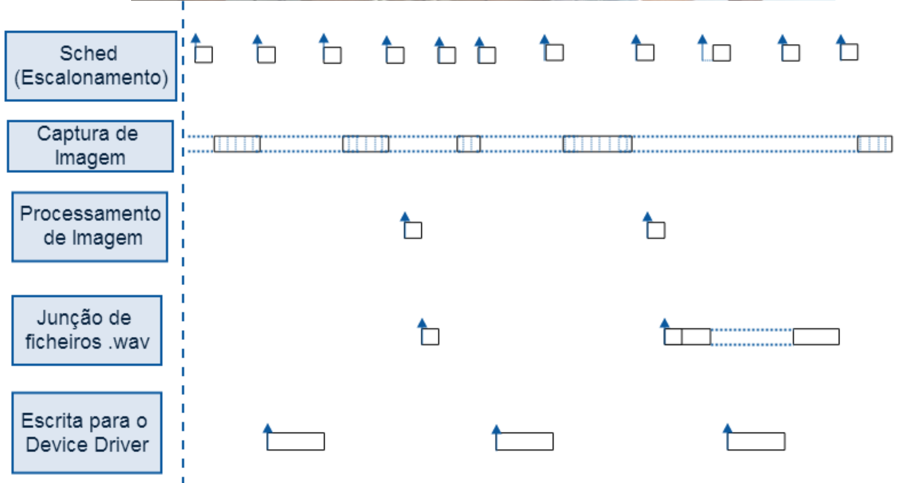


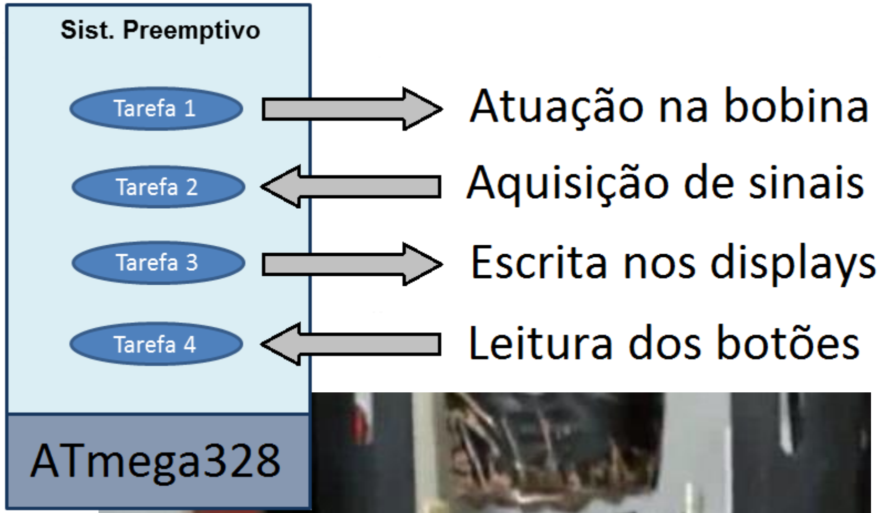


Examples of projects

- **Magnetic levitation device**
- **Drum machine**
- **Guitar tuner**







Sist. Preemptivo

Tarefa 1 → Atuação na bobina

Tarefa 2 ← Aquisição de sinais

Tarefa 3 → Escrita nos displays

Tarefa 4 ← Leitura dos botões

ATmega328