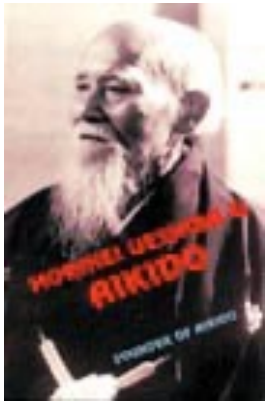
COMPSYS723 – Embedded Systems Design

LS1 - Introduction

Zoran Salcic
University of Auckland, New Zealand
Semester 1, 2023

Torimakishi
yari no hayashi ni
iru toki wa
kotate wa onoga
kokoro tozo shire

When surrounded
by a forest
of spears,
know that you must use
your own mind as a shield



Morihei Ueshiba, O Sensei
(The Great Teacher 1883-1969)

This Course Covers

- General aspects of design of embedded systems - design flow and design steps from specification to implementation of embedded systems
- Focuses on software-based embedded systems (or software component of systems that have HW/SW implementation)
- Explores concurrency as the basic vehicle to deal with and tame complexity of design
- Introduces methods for safe use of concurrency through synchronisation and communication of concurrent behaviours (tasks, threads, processes) and scheduling as the solution to guaranteed performance
  - via Real-Time Operating System as an extension to traditional programming languages used for design/specification
  - via concurrent programming languages and formal models of computation (MoC)

This Course Covers (cnt'd)

- The course divided into two parts with small-scale examples/projects abstracted from real-life systems
- Labs precede the assignments/projects
- Each assignment is worth 15% of overall marks- done in groups of two
- Final exam 70% of overall marks
- Prerequisite: COMPSYS 303 or equivalent knowledge acquired through other courses (that include proficiency and knowledge in using C/C++ programming for embedded platforms, as well as basic knowledge of interfacing of microprocessors with physical world and communications with physical world
- Lecturers: Zoran Salcic (Part I) and Avinash Malik (Part II)

Introduction Outline

- What are embedded/cyber-physical systems (ES/CPS)
- Typical applications and examples
- Underlying technology trends
- Implementation challenges
- Categorisation of embedded systems
- Research challenges

Embedded Systems – What they are and why are they important?

- An embedded system
  - employs a combination of hardware & software (a “computational engine”) to perform a specific function
  - is part of a larger system that may not be a “computer”
  - works in a reactive and time-constrained environment
  - is not meant to be programmed by a user
- Often referred to as Cyber-Physical Systems (CPS) to emphasise their interaction with physical world

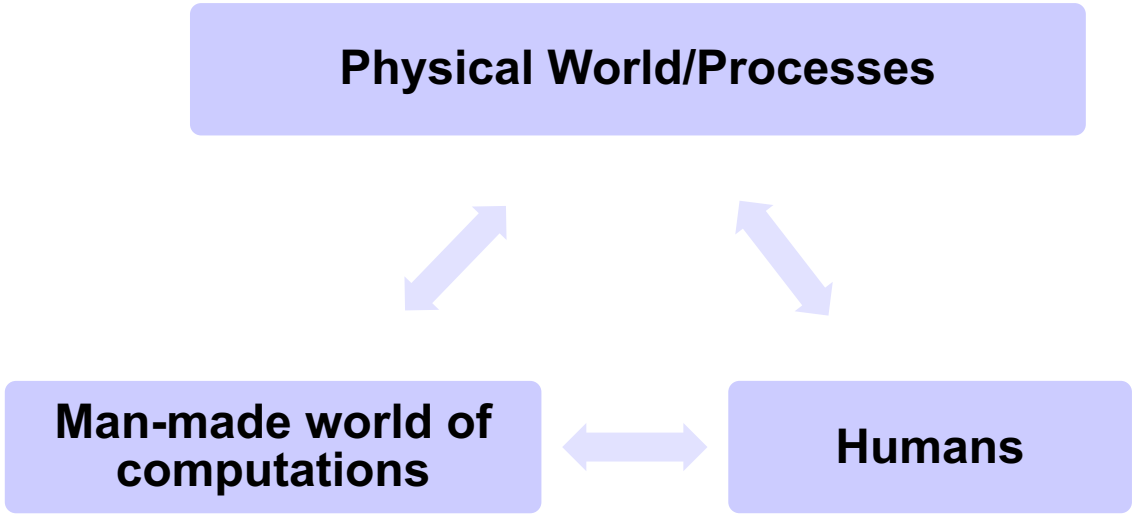
Embedded Systems – What they are and why are they important?

- Being key of mechanical and mechatronic devices (as a third industrial revolution) in 1990s and early 2000s
- Now, embedded systems are the key and foundation for creation of Internet of Things (IoT) and fourth industrial revolution (Industry 4.0 or Industrial Internet of Things, IoIT, and Industry 5.0)
- Also, key component of all kinds of “intelligent” systems

- Software is used for providing features and flexibility
- Hardware is used for performance & sometimes security

Cyber-Physical Systems (CPS)

- Symbiosis of physical world with man-made world of computing that may include interaction with humans
- New “term” for well known things
- Humans wish to change and control physical world (mechanical, chemical, biological) processes go back several thousands years



CPS - Key Developments and Enablers

What gave CPS new momentums in the last 50 years

- Developments in electronics/computing (since early 1970s)
- Densely integrated electronic systems: systems on chip (SoC), low power consumption at very low cost (since late 1990s)
- New sensing techniques: range of sensors for physical, chemical, environmental, bio sensing, etc
- Communication technologies: advances in wireless and wired networks (in 2000s)
- New Internet: beyond traditional - ambient intelligence, active, sensitive and reactive systems

Facts

- A typical embedded device has both hardware and software
  - Hardware components are microprocessors, digital signal processors, HW implemented (ASIC, VLSI, FPGA) application-specific functionalities and communication components
  - Software consists of applications that perform dedicated tasks and may include an embedded operating system (EOS) or a real-time operating system (RTOS), but also on bare-bone hardware without any OS
  - EOS/RTOS is responsible for performing and controlling dedicated tasks and steers applications in predictable and consistent manner
- Applications need to satisfy functional and non-functional (e.g. low power consumption, security, reliability, safety etc) requirements
- Systems on Chip (SoC) are key technological advance for many successful embedded systems solutions

Embedded Systems – Common Characteristics

- Application specific – made for a specific application
- Use of digital signal processing – process signals in digital form
- Reactive – react on the events from external/physical world
- Real-time – must perform functions within pre-defined time constraints
- Communicative – communicate with other systems
- Distributed – are part of spatial/geographically distributed system
- Man-machine interface – enable communication with humans

Specifics of Embedded Systems Engineering

- The design of embedded systems draws upon several disparate disciplines in EE and CS creating CSE (Computer Systems Engineering)
- Application domains (Signal processing, interfacing with physical world)
- Software engineering (Programming Languages, compilers, RTOS)
  - for implementing software components
- Complex digital design
  - for implementing the custom and semi-custom hardware components

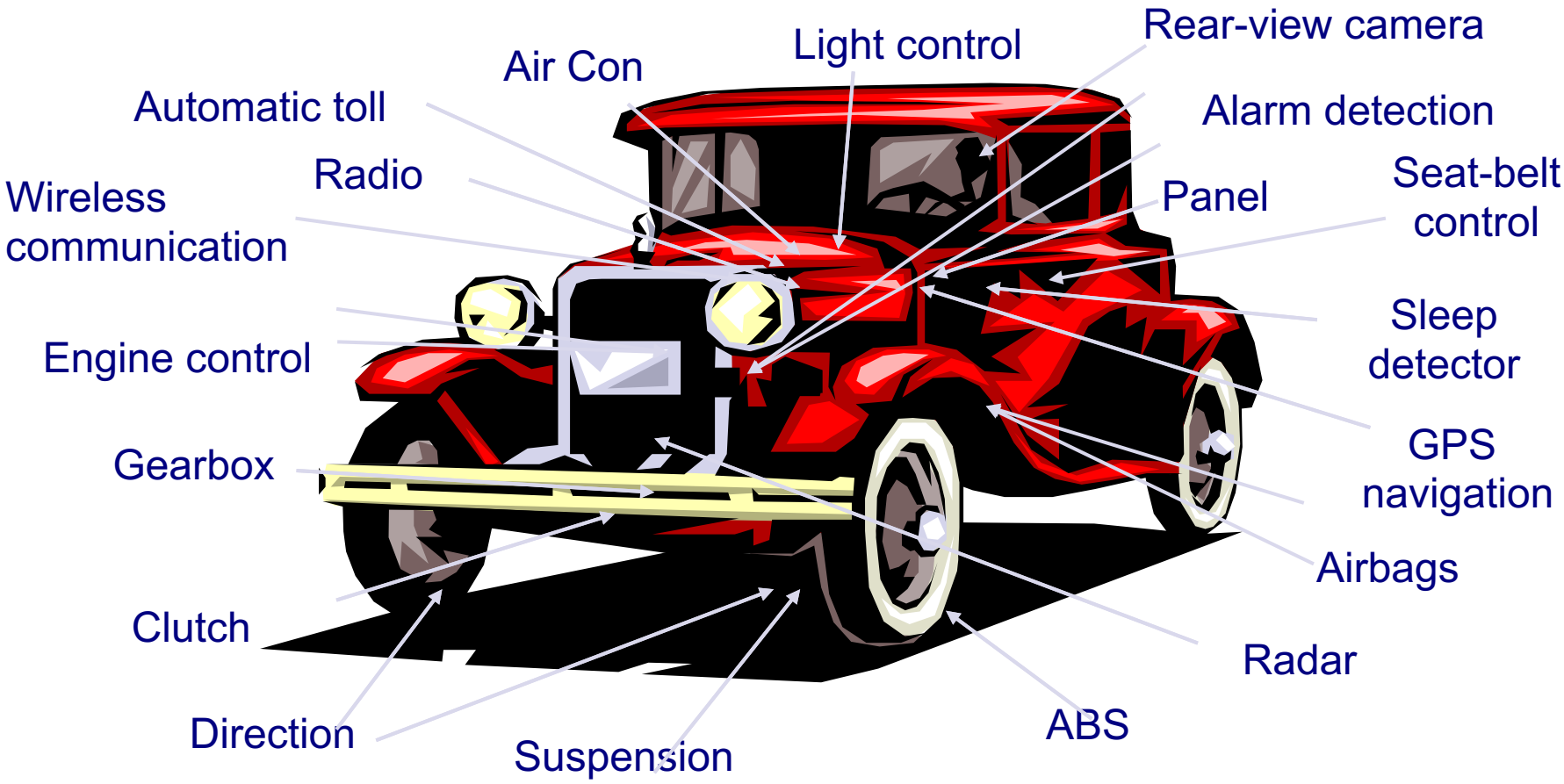
- Parallel/Distributed system design since many embedded systems are structured as a network of communicating processes and machines
- Real-time systems (hard- & soft- real time systems)

Embedded Systems are Application Driven

- Wireless systems (cellular phones, wireless modems, wireless sensor networks)
- Data communications systems (routers, switches, phones,…)
- Vehicle automation (tens of microprocessors in a typical vehicle) – highly structured networked systems
- Intelligent sensors (temperature, pressure, motion,…)
- Home and building networking and automation (appliances, security, entertainment, climate control, access control, etc)
- Control and instrumentation (adaptive and self-tuning controllers, fuzzy controllers,…)
- Entertainment (games, audio, video,…)
- and many more….

Modern Vehicle Example

This is not all…> 50 computers on board



Modified from original: Gerard Berry, Chief Scientist, Esterel Technologies

Application Characteristics

- Interact with their immediate physical environment
  - thus, cyber-physical

- Increasingly sophisticated and more complex
- Design strategies require familiarity and knowledge of applications
- Fixed functionality - determined by the system's interaction with its environment
- Opposite from general computing systems that have to support various applications and scenarios (e.g. personal computers)

Need for new computer architectures

- Execute diverse application functions such as
  - Digital signal processing algorithms (arithmetic operations)
  - Communication functions (protocols, state machines)
  - Reaction on random events in environment (activated randomly, state machines)
  - Control algorithms (arithmetic operations)
  - Machine inference
  - User interface (computer graphic, voice/sound interface)
- Hardware/software co-design has emerged as a basis for embedded custom computing machines tailored for the applications
- Architectures - combination of software-implemented and fixed hardware-implemented functions
- Heterogeneous architectures implemented on a single chip (SoC)

Embedded Systems: From Present to Future

Technology as a Driving Factor

- Technology enables integration of huge number of transistors on a single chip (tens of billions in extreme case)
- Operation frequencies don't increase and are becoming limiting factor
- Mixing analogue and digital circuits on a chip (required for interfacing with the physical world)
- Low power requirements (energy consumption is skyrocketing, need for energy-friendly systems)
- New extremely complex applications emerge and become possible (autonomous driving, real use of AI in real-time big data processing…)

Key issue: Complexity of Solutions

How to Describe/specify, synthesise and verify solutions

Examples of Emerging Applications

- Very-high speed encrypted communications for audio and video
- Wireless systems (5G and beyond) with picocells and picoradio, wi-fi, wireless sensor networks (WSN), combined with high computing power – software defined systems
- Internet of Things (IoT) and Industrial IoT (IIoT)
- Smart homes with wireless home networks
- Autonomous intelligent control (driving) systems
- Location-aware applications with high-precision positioning technology

Embedded Systems Categorisation

Two major types of embedded systems

- Reactive or control-dominated systems
  - Data and events arrive in irregular intervals and must be processed within strict timing constraints
- Data-flow or data-dominated systems
  - Data arrives in regular streams (usually as the result of periodic sampling or periodic delivery)
  - Must be processed in algorithms using basic arithmetic operations with required range and accuracy
- Modern systems are mixture of two types, hence we call them

Heterogeneous Embedded Systems

Reactive (Control-dominated) Systems

- They respond to incoming events/stimuli from environment by changing its internal state and producing output results
- They support a set of modes and settings and their real-time constraints are in the range of milliseconds
- Physical environment is delay intolerant; interaction with the environment often leads to non-determinism in execution
- Examples: lift controllers, home appliances (microwaves, dishwashers,…), vehicles (engine control, fuel injection, ABS,…), robotics applications, industrial automation, smart grid,…

Data-dominated Systems

- Also called transformational systems
- They are often real-time systems executing a(the) special function(s) within predefined time window(s)
- More powerful microprocessors, DSPs, ASICs or reconfigurable systems are required; even multi-core systems with hundreds of cores
- Most often dynamic synchronous mode of computation is applicable with a high presence of digital signal processing
- Examples: audio (MP3, MP4), video (MPEG2 and MPEG4, DVD, UHDTV, video-phone,…), consumer electronics (office equipment, video games,…), wireless communications (GSM, CDMA, DECT,3-5G…), telecommunications (ATM, ISDN, modems,…)

Data-dominated Systems

- Often characterised by the required sampling rate:
  - Telecommunication systems typically require from 10^4 to 10^6 samples per second
  - Typical video algorithms require 10^7 to 10^8 samples per second – they require 1-10 billion operations per second, and a bandwidth of more than 1Gbyte/s
- Embedded systems typically implemented as heterogeneous (architectures) systems consisting of dedicated and programmable parts
  - Dedicated part = hardware in ASIC or FPGA
  - Programmable part = software on DSP or GP processor core
  - Dedicated part + Programmable part => System-on-Chip (SoC)
- Designing such a system = Hardware/software Co-design

Embedded Systems – Commonalities

- Heterogeneity
  - Components (software, electronics, mechanical components, optics, etc.)
  - Types of data processing (data-driven vs control-driven)
  - Implementation architectures (multiple processors on chip, SoC, NoC)
- Complexity (complex components, real-time requirements, low power, reliability, security)
- Energy autonomy
  - Alternative energy sources (vibration, light, walking,…)
- Huge design complexity if designed from scratch (number of transistors, number of lines of code, inter-relationships)

Embedded Systems - Properties

- Must be **dependable**,
  - **Reliability *R(t)* =** probability of system working correctly provided that is was working correctly at *t*=0
  - **Maintainability *M(d)* =** probability of system working correctly *d* time units after error occurred.
  - **Availability *A*(*t*)**: probability of system working at time *t*
  - **Safety**: no harm to be caused to people and/or assets
  - **Security**: confidential and authentic communication
- Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong.
- Making the system dependable must not be an after-thought, it must be considered from the very beginning

Embedded Systems - Properties

- Must be **efficient**
  - Energy efficient
  - Code-size efficient
    (especially for systems on a chip)
  - Run-time efficient
  - Weight efficient
  - Cost efficient
- **Dedicated** towards a certain **application**
  Knowledge about behavior at design time can be used to minimize resources and to maximize robustness
- **Dedicated user interface**
  (often no mouse, keyboard and screen)
- **Hybrid systems** (analog + digital parts).

Embedded Systems - Properties

- Many ES must meet **real-time constraints**
  - A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.
  - For real-time systems, right answers arriving too late are wrong.
  - **„A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe“** [Kopetz, 1997].
  - All other time-constraints are called **soft**.
  - A guaranteed system response has to be explained without statistical arguments
- Usually **connected to physical environment** through sensors and actuators, cyber-physical systems

Challenges and Solutions

Challenges

- **Complexity** - system design involves the design of integrated computers and programs (software) at the same time
  - thousands, even millions, of elements in system specification
- **Bugs (errors)** are the biggest enemy!

Solutions

- Better specifications - new system level languages
- Better synthesis – based on formal models of computation and with automatic synthesis
- Better formal verification - property checking

System architects and designers start understanding the value of **system-level design**

Research Challenges

- Systems will communicate and interact in ways that were unforeseen during their design
- Safety and security become very important issues
- Third parties will contribute components and subsystems – IP re-use (compatibility, trust)
- During the system lifetime, parts of the software and/or hardware will be changed and unknown software will be downloaded (trusted software,…)
- Ability to communicate (networked, connected, unobtrusively communicating; with their environment typically by means of sensors and actuators)
- Decentralized control, adaptive behavior, self configuring, and self restoring

Research Challenges

- Model-based specification and simulation – formal approach (formal verification of properties)
- Automatic mapping from specification languages onto target architectures (hardware and/or software platforms) - automated design flows – HW/SW co-design
- Integration of multiple technologies (semiconductors, wireless communications, mechanical, optical, chemical,….)
- Early integration of software and testing and validation