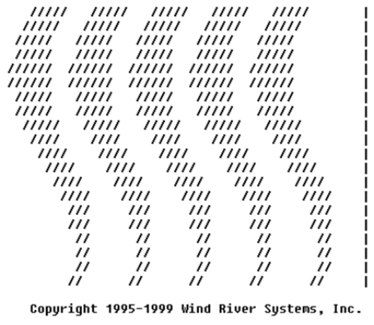
FYS 4220/9220 – 2011 / #1

Real Time and Embedded Data Systems and Computing

Introduction and baseline concepts


```
T O R N A D O

Development System

Host Based Shell

Version 2.0.2

Copyright 1995-1999 Wind River Systems, Inc.
```

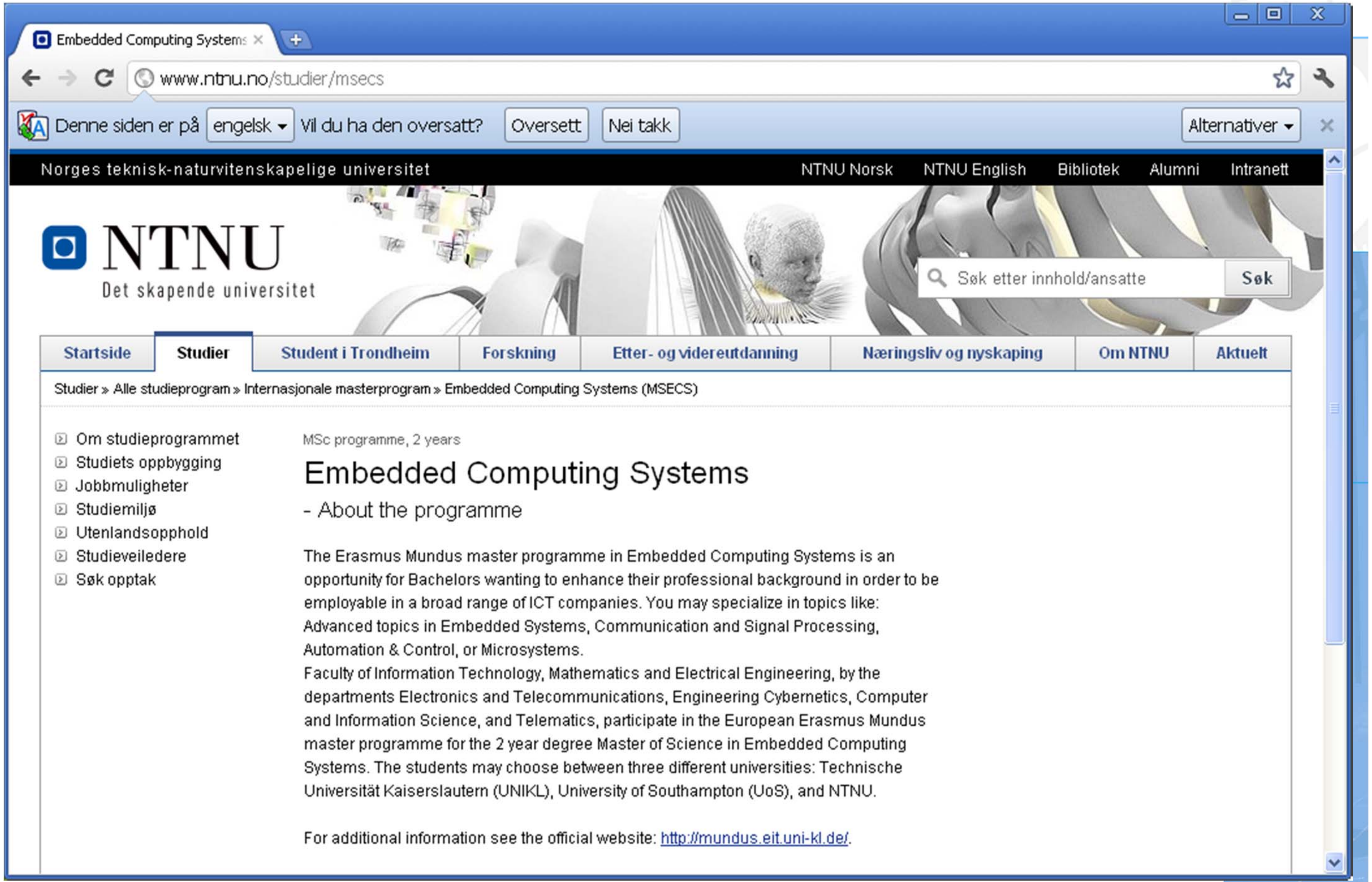
T.B. Skaali, Department of Physics, University of Oslo)

FYS 4220 / 9220 – 2011 version

- **In 2010 the course was revised!**
  - Content: approximately 50/50 programmable logic (VHDL) / Real-Time – Embedded systems
    - VHDL lectures and lab course given by Jan Kenneth Bekkeng. See description and information given by Bekkeng
    - Real-Time / Embedded part given by Bernhard Skaali
- **Real-Time and Embedded part:**
  - Impossible to find a single text book which covers the curriculum. Therefore: the lectures are based on several sources and own contributions. **Curriculum ≡ lectures plus lab exercises**
  - Background required (wanted):
    - Basic knowledge of computer electronics (FYS3230, FYS3240, IfI)
    - Some programming knowledge in C / C++
  - Teaching:
    - Lectures
    - Lab exercises on VME single board processors running the VxWorks Real-Time Operating System (RTOS)

Some literature references

- [Ref. 1] Phillip A. Laplante, ”Real-Time Systems Design and Analysis”, Third Edition, John Wiley & Sons / IEEE Press, 2004
  - Dr. Laplante has more than 20 years experience in building, studying and teaching real-time systems as well as long interactions with NASA, Lockheed-Martin, MIT, etc.
  - http://www.personal.psu.edu/pal11/
- [Ref. 2] Alan Burns and Adny Wellings, ”Real-Time Systems and Programming Languages, Third Edition – Ada 95, Real-Time Java and Real-Time POSIX, Addison-Wesley, 2001 [B/W]
  - http://www.cs.york.ac.uk/rts/books/RTSBookThirdEdition.html
- [Ref. 3] ”Real-Time Systems”, The International Journal of Time-Critical Computing Systems, Springer
  - http://www.springer.com/computer/communication+networks/journal/11241
- [Ref. 4] Embedded systems:
  - http://www.eetimes.com/design/embedded
  - http://www.eejournal.com/design/embedded


Embedded Computing Systems
www.ntnu.no/studier/msecs
Denne siden er på engelsk Vil du ha den oversatt? Oversett Nei takk Alternativer
Norges teknisk-naturvitenskapelige universitet
NTNU Norsk NTNU English Bibliotek Alumni Intranett
NTNU
Det skapende universitet
Søk etter innhold/ansatte Søk
Startside Studier Student i Trondheim Forskning Etter- og videreutdanning Næringsliv og nyskaping Om NTNU Aktuelt
Studier » Alle studieprogram » Internasjonale masterprogram » Embedded Computing Systems (MSECS)
Om studieprogrammet
Studiets oppbygging
Jobbmuligheter
Studiemiljø
Utenlandsopphold
Studieveiledere
Søk opptak

MSc programme, 2 years

Embedded Computing Systems

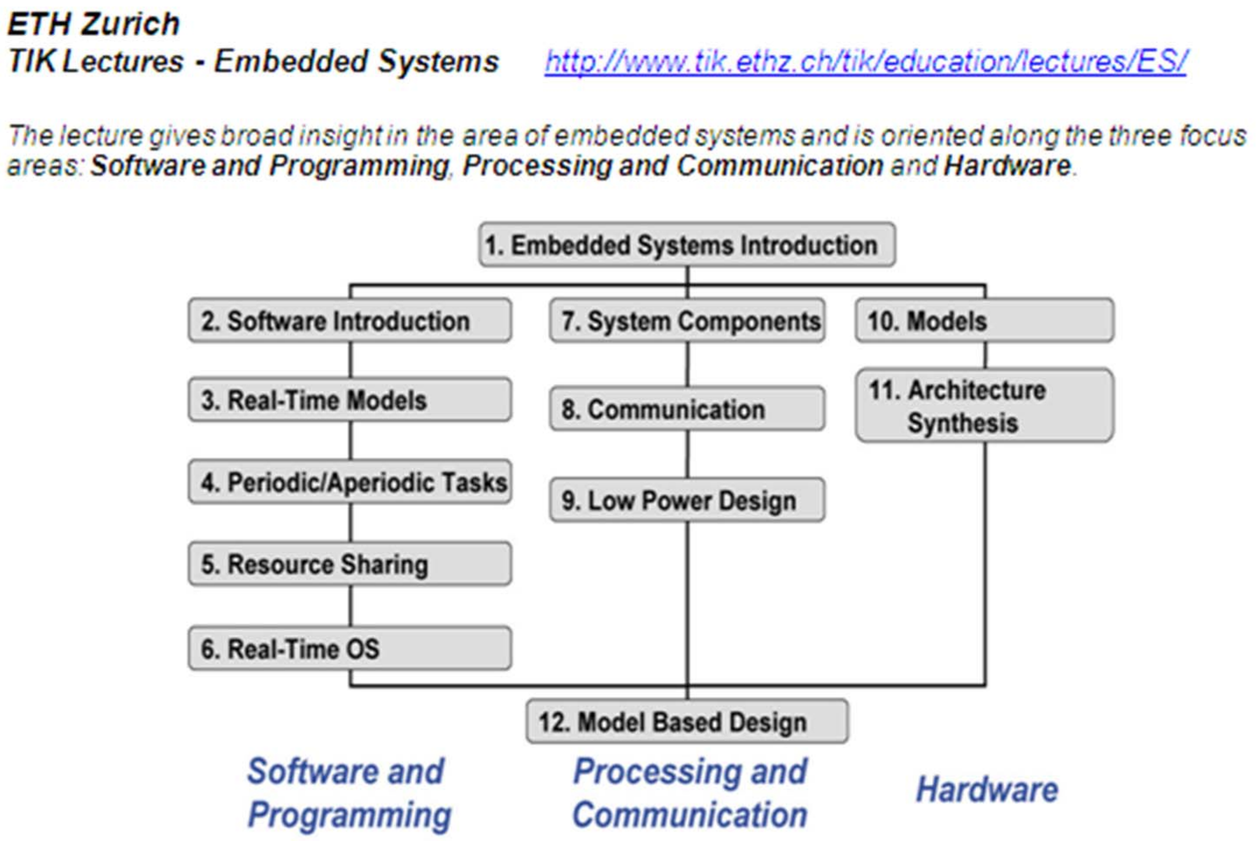
- About the programme

The Erasmus Mundus master programme in Embedded Computing Systems is an opportunity for Bachelors wanting to enhance their professional background in order to be employable in a broad range of ICT companies. You may specialize in topics like: Advanced topics in Embedded Systems, Communication and Signal Processing, Automation & Control, or Microsystems.
Faculty of Information Technology, Mathematics and Electrical Engineering, by the departments Electronics and Telecommunications, Engineering Cybernetics, Computer and Information Science, and Telematics, participate in the European Erasmus Mundus master programme for the 2 year degree Master of Science in Embedded Computing Systems. The students may choose between three different universities: Technische Universität Kaiserslautern (UNIKL), University of Southampton (UoS), and NTNU.

For additional information see the official website: http://mundus.eit.uni-kl.de/.

UNIVERSITY OF OSLO

Embedded systems courses at ETH Zurich

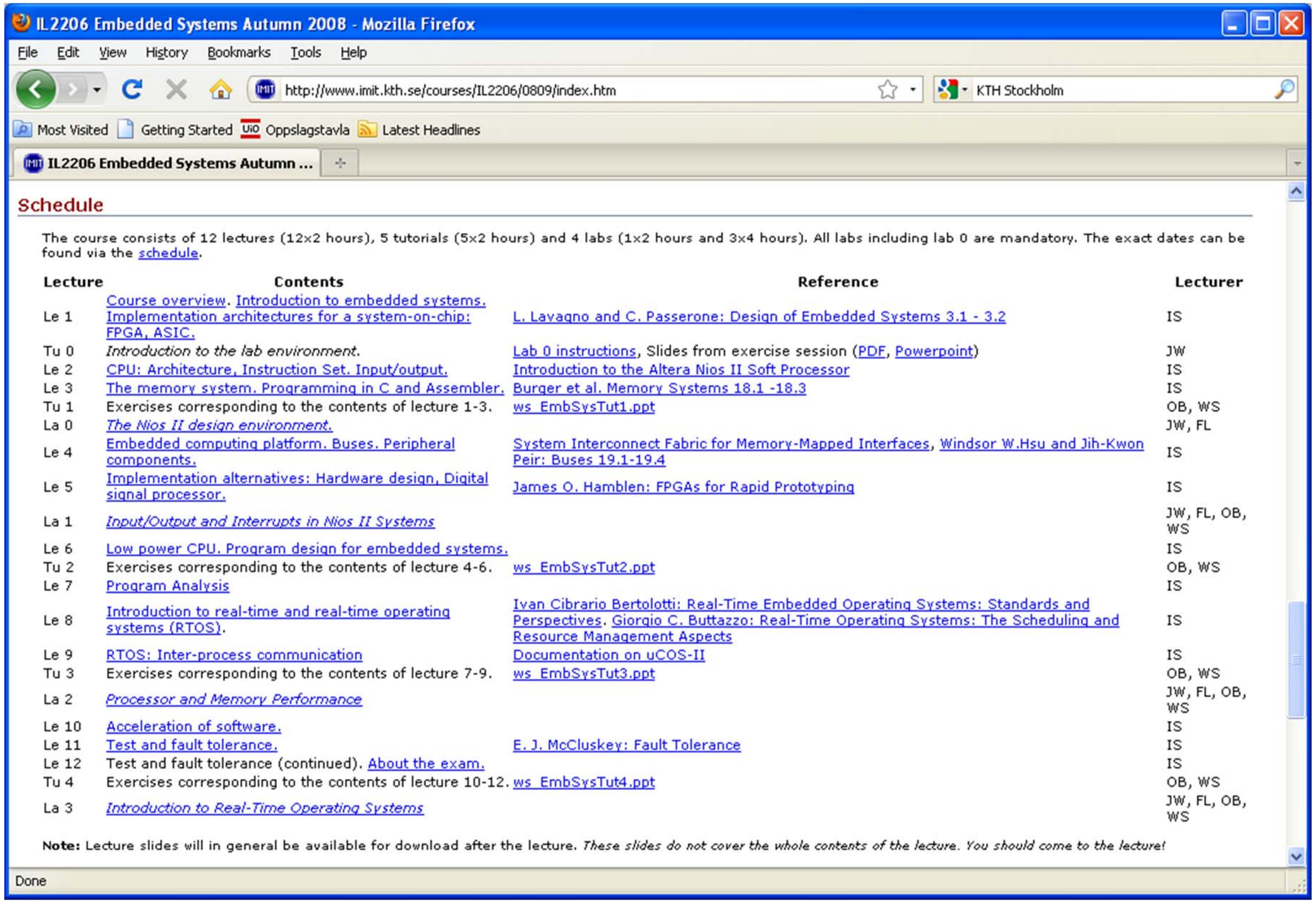


ETH Zurich
TIK Lectures - Embedded Systems http://www.tik.ethz.ch/tik/education/lectures/ES/

*The lecture gives broad insight in the area of embedded systems and is oriented along the three focus areas: **Software and Programming**, **Processing and Communication** and **Hardware**.*

UNIVERSITY OF OSLO

Embedded systems course at KTH Stockholm


IL2206 Embedded Systems Autumn 2008 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.imit.kth.se/courses/IL2206/0809/index.htm
KTH Stockholm
Most Visited Getting Started Oppslagstavla Latest Headlines
IL2206 Embedded Systems Autumn ...
Done

Schedule

The course consists of 12 lectures (12x2 hours), 5 tutorials (5x2 hours) and 4 labs (1x2 hours and 3x4 hours). All labs including lab 0 are mandatory. The exact dates can be found via the schedule.

Lecture	Contents	Reference	Lecturer
Le 1	Course overview. Introduction to embedded systems. Implementation architectures for a system-on-chip: FPGA, ASIC.	L. Lavagno and C. Passerone: Design of Embedded Systems 3.1 - 3.2	IS
Tu 0	*Introduction to the lab environment.*	Lab 0 instructions, Slides from exercise session (PDF, Powerpoint)	JW
Le 2	CPU: Architecture, Instruction Set. Input/output.	Introduction to the Altera Nios II Soft Processor	IS
Le 3	The memory system. Programming in C and Assembler.	Burger et al. Memory Systems 18.1 -18.3	IS
Tu 1	Exercises corresponding to the contents of lecture 1-3.	ws_EmbSysTut1.ppt	OB, WS
La 0	*The Nios II design environment.*		JW, FL
Le 4	Embedded computing platform. Buses. Peripheral components.	System Interconnect Fabric for Memory-Mapped Interfaces, Windsor W.Hsu and Jih-Kwon Peir: Buses 19.1-19.4	IS
Le 5	Implementation alternatives: Hardware design, Digital signal processor.	James O. Hamblen: FPGAs for Rapid Prototyping	IS
La 1	*Input/Output and Interrupts in Nios II Systems*		JW, FL, OB, WS
Le 6	Low power CPU. Program design for embedded systems.		IS
Tu 2	Exercises corresponding to the contents of lecture 4-6.	ws_EmbSysTut2.ppt	OB, WS
Le 7	Program Analysis		IS
Le 8	Introduction to real-time and real-time operating systems (RTOS).	Ivan Cibrario Bertolotti: Real-Time Embedded Operating Systems: Standards and Perspectives. Giorgio C. Buttazzo: Real-Time Operating Systems: The Scheduling and Resource Management Aspects	IS
Le 9	RTOS: Inter-process communication	Documentation on uCOS-II	IS
Tu 3	Exercises corresponding to the contents of lecture 7-9.	ws_EmbSysTut3.ppt	OB, WS
La 2	*Processor and Memory Performance*		JW, FL, OB, WS
Le 10	Acceleration of software.		IS
Le 11	Test and fault tolerance.	E. J. McCluskey: Fault Tolerance	IS
Le 12	Test and fault tolerance (continued). About the exam.		IS
Tu 4	Exercises corresponding to the contents of lecture 10-12.	ws_EmbSysTut4.ppt	OB, WS
La 3	*Introduction to Real-Time Operating Systems*		JW, FL, OB, WS

Note: Lecture slides will in general be available for download after the lecture. *These slides do not cover the whole contents of the lecture. You should come to the lecture!*

UNIVERSITY OF OSLO

What is Real-Time and Embedded Computing?

- *In computer science, **real-time computing** (RTC), or "reactive computing", is the study of hardware and software systems that are subject to a "real-time constraint"—i.e., operational deadlines from event to system response. By contrast, a non-real-time system is one for which there is no deadline, even if fast response or high performance is desired or preferred.*
- *The needs of real-time software are often addressed in the context of **real-time operating systems**, and **synchronous programming languages**, which provide frameworks on which to build real-time application software. (Ref. Wikipedia)*

What is Real-Time and Embedded Computing?

- *An **embedded system** is a computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts.*

  Embedded systems span all aspects of modern life and there are many examples of their use: mobile phones, PDAs, MP3 players, digital cameras, GPS receivers, household appliances, avionics system, medical systems, etc. Embedded processors can be microprocessors or microcontrollers. The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard and/or screen. (Ref. Wikipedia)

  Many universities are incorporating embedded system concepts into their electronic/informatics science curricula, some examples are shown on the following pages.

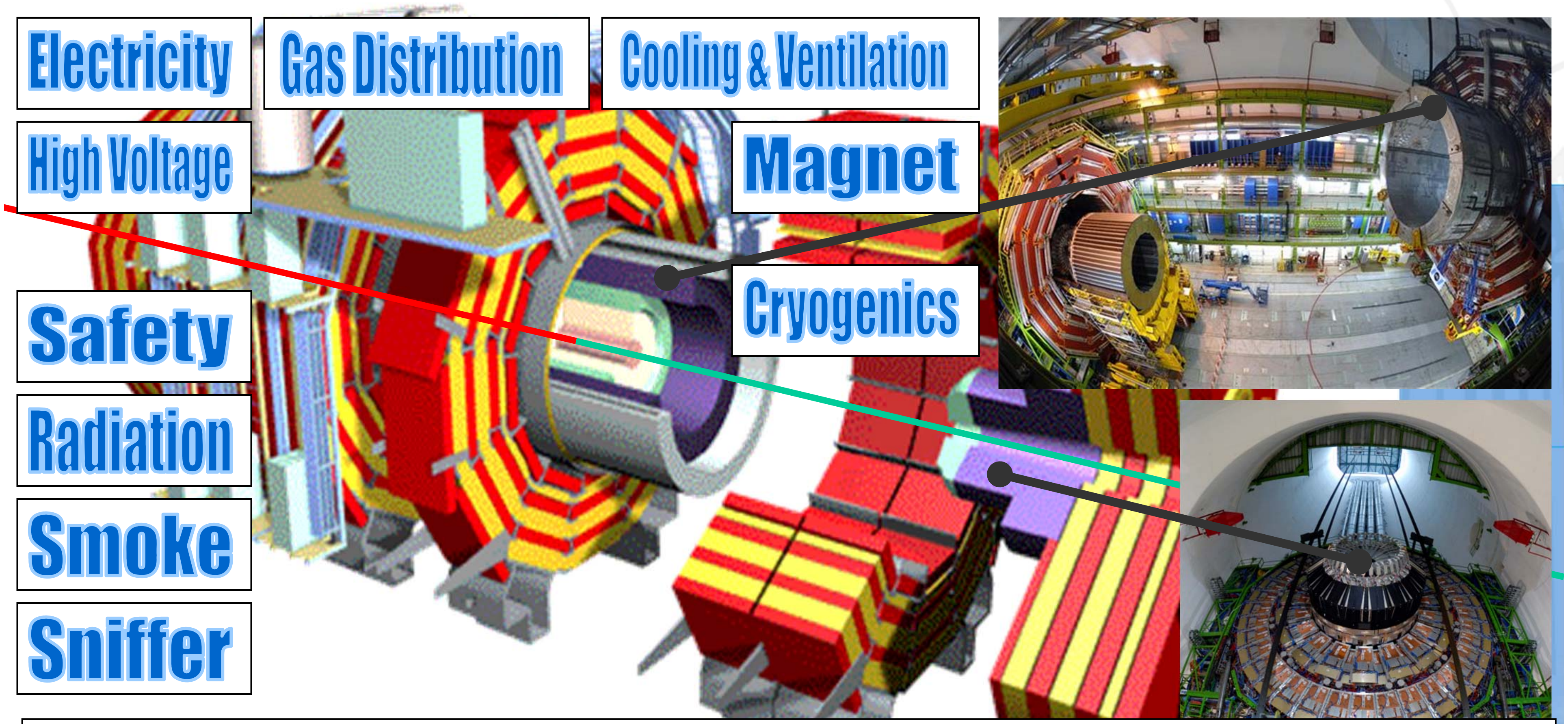
Embedded products are found in:

- Industry
- Automotive
- Aerospace
- Medical systems
- Mobile systems
- Communication
- Networking
- Household products (dishwasher, etc)
- Media products – broadcasting
- Cameras
- ---- in other words, everythere ----

A very large scale control system for LHC at CERN is shown on next page

UNIVERSITY OF OSLO

Control Systems for CERN LHC Experiments

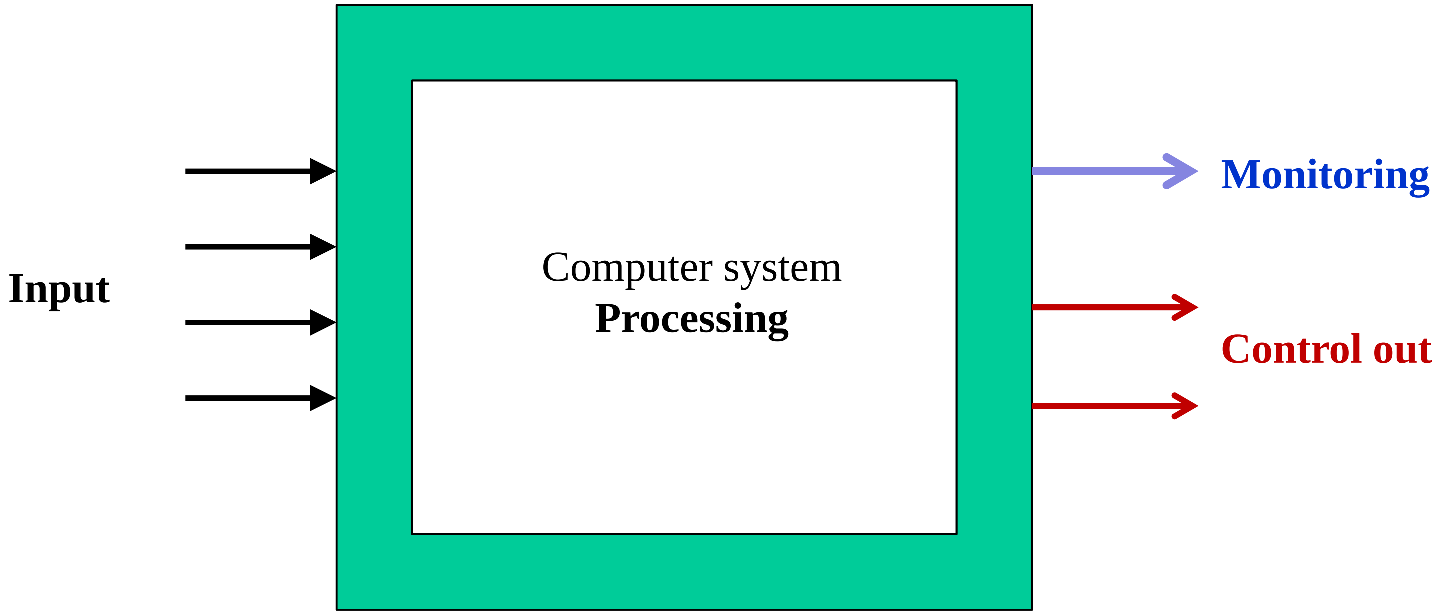


About one million control channels

TERMINOLOGY AND DEFINITIONS

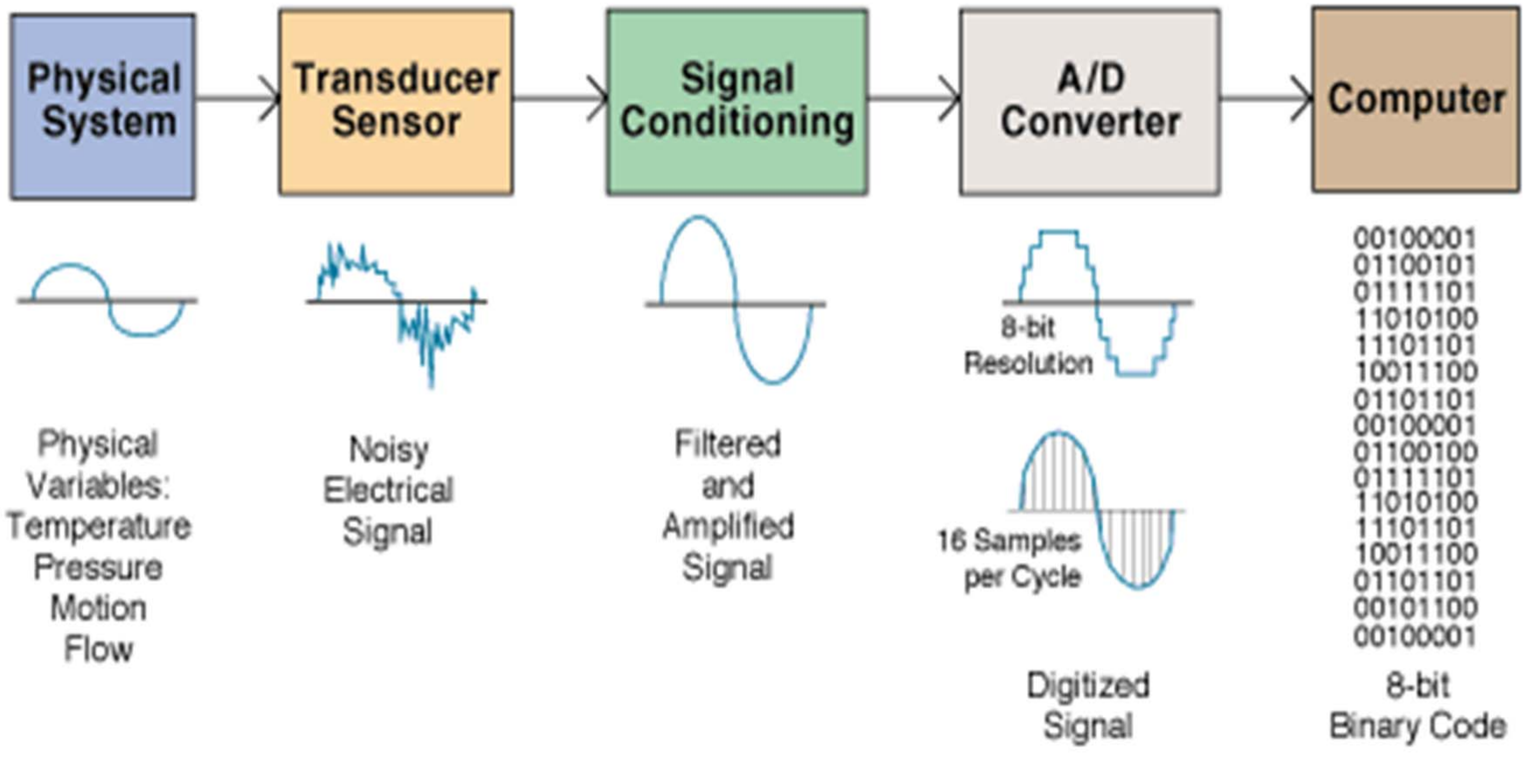
System concepts

- A **Real-time system** is a mapping of a set of inputs to a set of outputs according to given time constraints
  - In most applications, a Real-time system never terminates, it runs «for ever». However, the type and amount of input data, the algorithms, the number of processes, interprocess communication, interrupt frequency etc are numerous. This makes a realistic analysis of a Real-time system very difficult. Simulation can be an important tool.



UNIVERSITY OF OSLO

Case: signal processing in a Data Acquisition system



Schematic 1: Data Acquisition System

UNIVERSITY
OF OSLO

Some basic definitions borrowed from Laplante [Ref. 1]

The time between the presentation of a set of inputs to a system (stimulus) and the realization of the required behaviour (response), including the availability of all associated outputs, is called the response time of the system

A Real-time system is a system that must satisfy explicit (bounded) response-time constraints or risk severe consequences, including failure.

A failed system is a system that can not satisfy one or more of the requirements stipulated in the formal system specification.

A Real-time system is one whose logical correctness is based on both the correctness of the outputs and their timeliness.

Characterisation of Real-time systems

Obviously there is a difference regarding possible consequences for i) a data acquisition system at a LHC experiment at CERN or ii) an aeroplane control system if a time constraint is not met. (In fact, for i) one knows that some data will be lost!).

Real-time systems are typically characterized as **hard**, **firm** and **soft**. The following definitions are taken from [Ref. 1].

A hard Real-time system is one in which failure to meet a single deadline may lead to complete and catastrophic system failure.

A firm Real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete and catastrophic system failure.

A soft Real-time system is one in which performance is degraded but not destroyed by failure to meet response time constraints.

Reliability, availability, safety and security

- Reliability, availability, safety and security are other important characteristics of Real-time systems
- In particular, for hard Real-time systems, one must guarantee high to extreme reliability.
- Safe
  - An absolute requirement where life or environment may be endangered (but still such systems fails from time to time…). However, a 100% safe aeroplane is the one which never flies!
- These aspects of Real-time systems will be discussed in a later lecture.

DESIGN AND IMPLEMENTATION ISSUES

Real-time computer systems

- Real-time and embedded systems are built using a very wide range of hardware, software, computer languages, sensors and instrumentation .
- The simplest system one can imagine is a single control loop on a microprocessor using polling.
- In this course we will discuss computer systems that support concurrent processing under control of a Real-time operating system (RTOS). Whether the computing hardware is a processor chip or a «soft-core» processor implemented in a FPGA is irrelevant.

Time, Events and Determinism [see also Ref.1]

As the physicists know, time always move forward. In a Real-time system the occurrence of an activity may be in absolute calendar time or relative to another activity, for instance an interrupt. The baseline for designing a Real-time system is to define and understand the basis for timing constraints.

External input as well as process activity in a computer system, say inter-process communication, will result in a change of the flow of the execution of a computer program.

Definition: An occurrence that causes the program counter to change non-sequentially is considered a change of flow-of-control and thus an **event**.

Definition: In scheduling theory (later lecture), the «release» time of a processing activity is similiar to an event. The release time is the time at which an instance of a scheduled task is ready to run, and is generally associated with an interrupt, either from a clock or from data.

Synchronous and Asynchronous Events [see also Ref.1]

Synchronous events are those that occur at predictable times in the flow-of-control. An example is a trap instruction that triggers the kernel.

Asynchronous events occur at unpredictable times and are usually caused by external interrupts.

Is a clock that gives an interrupt every 20 msec a synchronous event? It is a **periodic event**, but the point in time where its interrupt routine start to process depends on other activities in the computer, for instance if processing is active on a higher interrupt priority level. Hence, a clock driven event must be considered as asynchronous.

Events that do not occur at regular intervals are **aperiodic**. Aperiodic events that occur very infrequently are called **sporadic**.

<u>A major problem in scheduling Real-time processes is how to estimate the effect of non-periodic events.</u>

UNIVERSITY OF OSLO

COMMON MISCONCEPTIONS

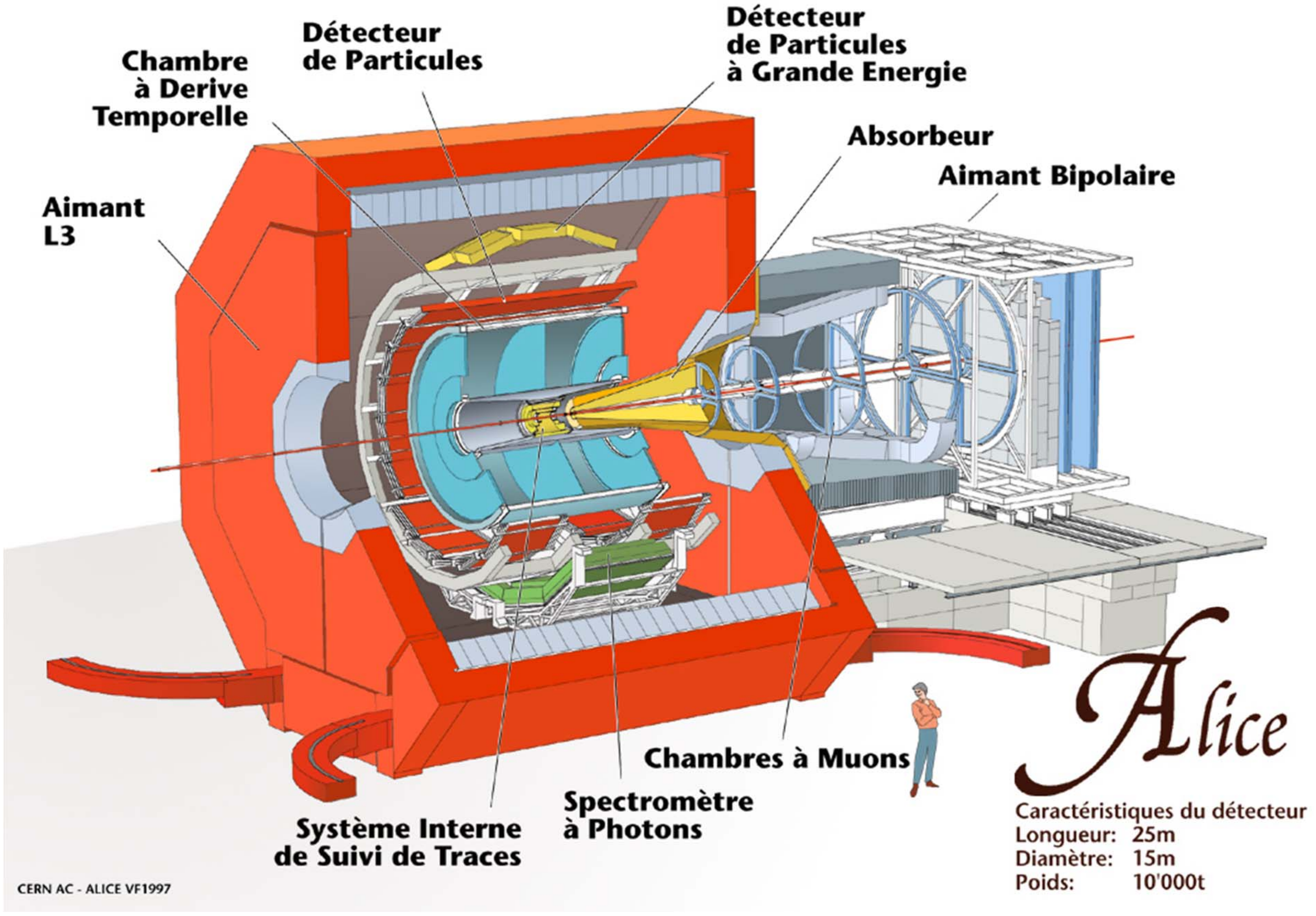
Some misconceptions as listed by Laplante (Ref. 1)

- Real-time systems are synonymous with “fast” systems!
  - True that many hard real-time systems must have response times of millisec or shorter. However, the deadline for a ticket reservation system is seconds.
- Rate-monotonic analysis has solved the “real-time problem”!
  - In a rate-monotonic system the process priorities are assigned according to the cyclic frequency of execution. Studied since the 70’s, easy to analyze, but represents an ideal environment seldom found in real life.
- There are universal, widely accepted methodologies for Real-time systems specification and design!
  - *“Even after more than 30 years of research there is no methodology available that answers all the challenges of real-time specification and design for all the time and for all applications”*
- There is no need to build a real-time operating system, because many commercial products exist!
  - Indeed, there are many excellent RT-OS’s, but for a small embedded system they may be an overkill.
- The study of real-time systems is mostly about scheduling theory!
  - Counting up scientific papers on Real-Time computing on may indeed get this impression!

CASE: DATA ACQUISITION SYSTEM AT CERN LHC EXPERIMENT

The ALICE detector at CERN/Large Hadron Collider

UNIVERSITY OF OSLO

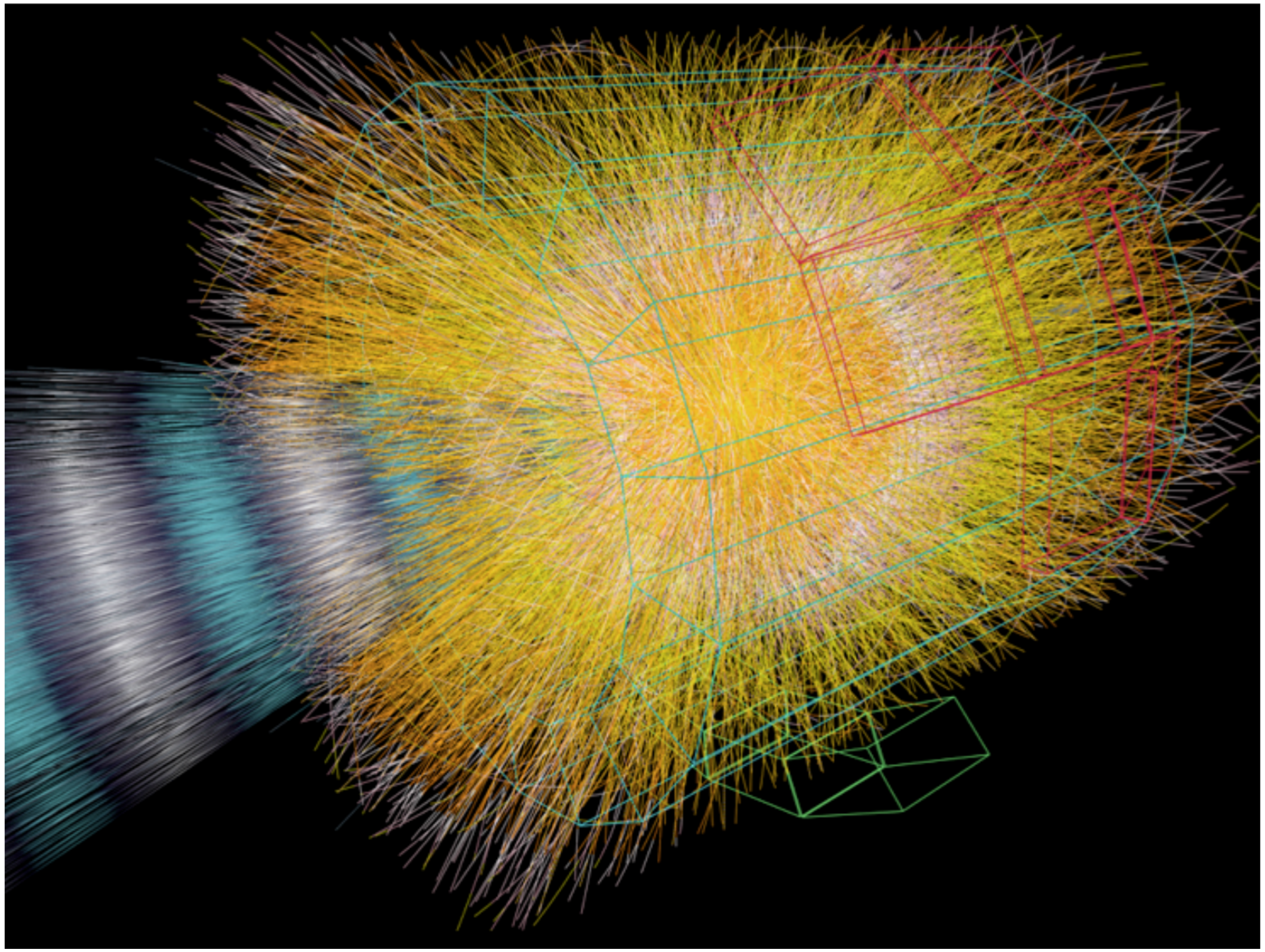


The ALICE detector

- ALICE is one of the four experiments at the circular Large Hadron Collider (LHC) at CERN. In the LHC protons or lead ions are accelerated to almost the speed of light and steered to collide at the center of the detectors. A collision creates an enormous energy density in a very small volume (”fireball”), which almost immediately is converted into radiation, see next page.
- The task of the sensors and data acquisition (DAQ) system is to measure the position, energy and type of radiation from the ”fireball”.
- A pictorial organization of the data stream is shown on the second page.
- The architecture of the ALICE DAQ data system is shown on the third page. A central component not shown is the ”Trigger system” that defines when data shall be read out from the front-end electronics by selecting the most interesting physics signals. The data streams from the top, showing how data records from the sub-detectors are merged to a complete data set (”event”) and written onto mass storage for off-line analysis. The labels stand for processing activities or buffer memories. The total number of processors is several thousand. The system interconnect is Gigabit Ethernet.

Simulated radiation after collision in LHC between two Pb nuclei

The radiation passes through detectors which register sufficient number of space points to reconstruct the type and path of the radiation. That implies a very large number of detector elements!



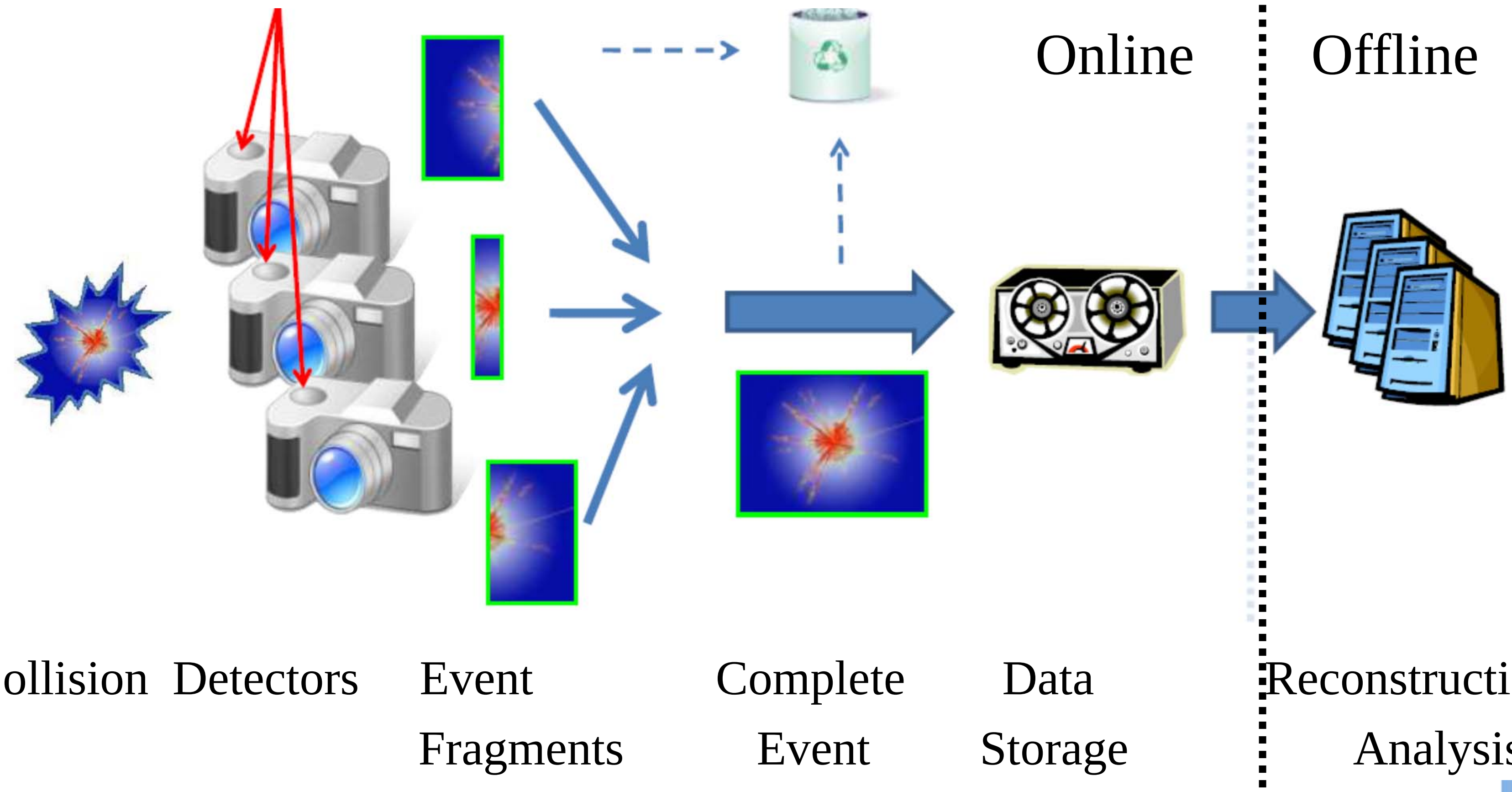
CERN Jan 2009

CERN Trigger -> Data Acquisition -> Offline

UNIVERSITY OF OSLO

Ultrafast processing of selected data in the trigger system decides whether the data from a collision shall be read out or not. Only a few of the collisions give interesting ***physics*** *data!*

Trigger (decision) + Data Acquisition (action) – Processing



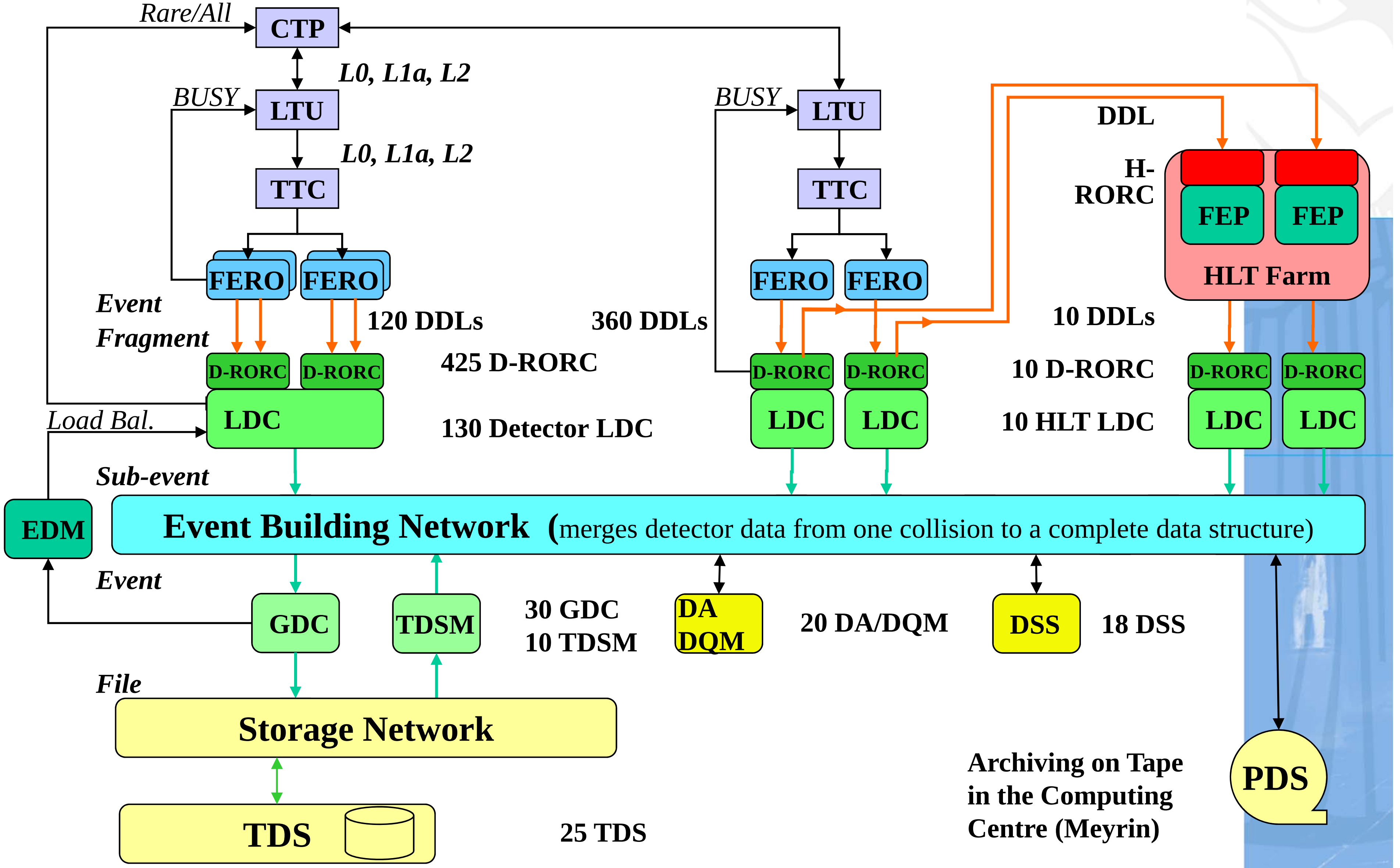
Drawing by S. Chapeland

CERN Jan 2009

CERN ALICE DAQ architecture

UNIVERSITY OF OSLO

A CERN LHC experiment has around 20 million electronic channels to read out



CERN Jan 2009

Concurrent activities

- The DAQ system illustrated on the previous pages is structured into a large number of **concurrent parallel activities**
  - Processing interrupts from the front-end electronics
  - Data buffer handling, for instance multiple buffering to avoid dead time
  - Moving data from buffers onto data links
  - Receiving data in the processors
  - Doing something with the data, filtering, formatting, etc
  - Writing processed data onto mass storage
  - Plus activities like monitoring, error handling, etc, etc
- Concurrency will be discussed in Lecture #2
- A **control system** also implements a feedback to the input process. Control algorithms are not covered in FYS 4220.

THE CONCEPT OF A PROCESS

The ”process”

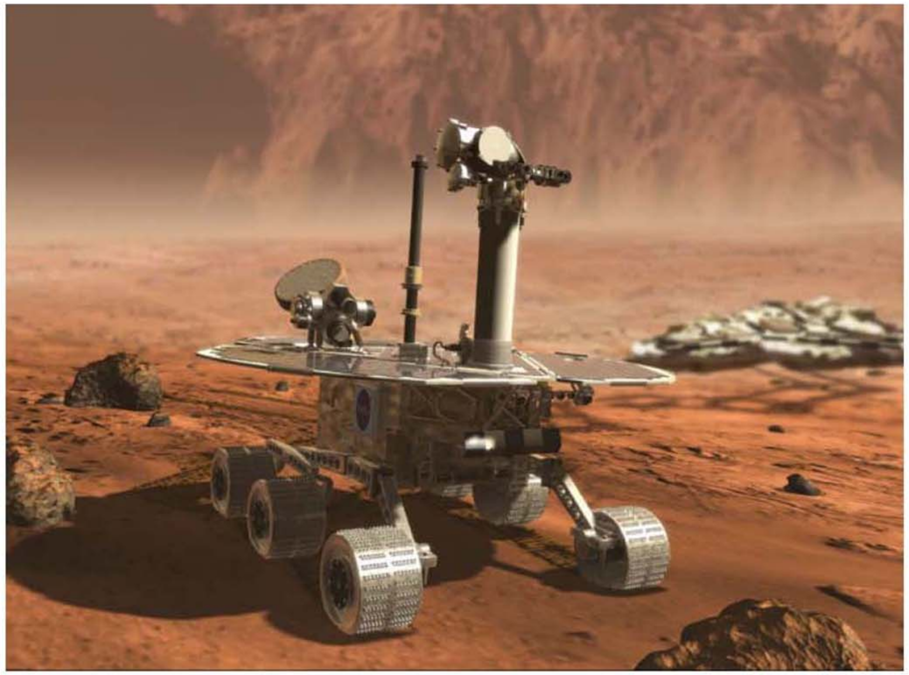
- Process: the execution of a running program
  - Logical unit scheduled and controlled by the operating system. Also called task in VxWorks.
    - Comprises the code, data, allocated resources, and a process descriptor
    - Interference with other processes only through system calls
      - However, in a system without memory protection, a process may have full access to other processes’ memory (”flat memory”)!
  - Thread (”tråd”): a subprocess within a process, sharing it’s resources.
    - Shorter context switching time for thread->thread than for process->process !
- The process descriptor, a data structure that
  - keeps the process attributes and information during its lifespan, like
    - Process attributes
    - Process relationships
    - Process memory space
    - File management
    - Communication facilities
    - Signal management
    - Resource limits
    - Scheduling related fields

REAL-TIME OPERATING SYSTEMS

Choice of RTOS for the lab exercises

- Why use VxWorks and not Linux?
  - VxWorks from Wind River is an industry leader in Real-Time systems
  - It offers a very wide choice of software components
  - Based on host – target configuration. System development is done on a host computer, typically a PC. The excutable code is downloaded to the target processor.
  - Excellent GUI (Graphical User Interface)
  - However, VxWorks is a rather expensive solution,
  - so, for systems which are not hard Real-Time, Linux is a very good, and much cheaper,choice
  - For more info on Wind River products, see http://www.windriver.com/
- Lab exercises
  - Implementation of some important kernel services to be discussed in the lectures

VxWorks on the Mars Exploration Rovers



UNIVERSITY OF OSLO

HARDWARE AND SOFTWARE

Computer architecture and busses

- FYS4220 is not a computer architecture course, but computing hardware and electronics are obviously integral components of Real-time and Embedded computing
- Elements of computer and bus hardware are therefore integrated into many of the lectures as well as the lab exercises
- You can not design and implement Real-time / Embedded systems without touching hardware! In fact, this is also great fun!

Software

- The choice of a programming language is an important parameter for building Real-time systems. In this course the C language is used in the lab exercises, for several reasons: i) C is a hardware-near language, ii) it is extensively used in industry, iii) the amount of code to be written is small, and iv) FYS4220 is not an Informatics course.
- Real-Time system design is a complex discipline in computer systems engineering, based on elements from *Control Theory, Programming Languages, Scheduling Theory, Algorithms, Queing Theory, Operating Systems, Software Engineering, Computer Architecture and Data Structures [Ref. 1]*. It is obvious that FYS4220 can not cover all these fields!