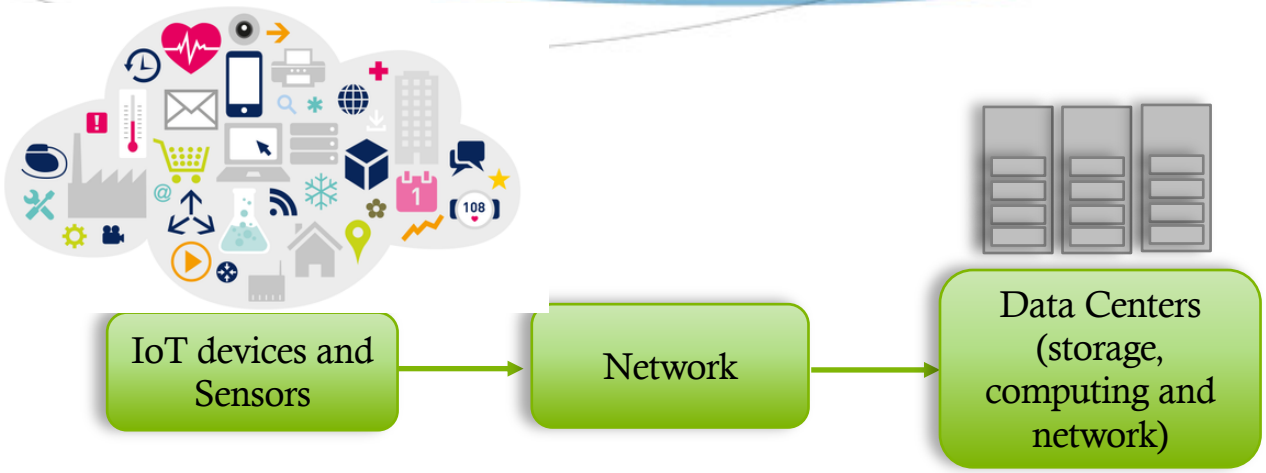
Embedded Systems in IoT Era

Eli Bozorgzadeh
Professor
Computer Science Department
University of California, Irvine

Data Processing in Big Data and IoT Era



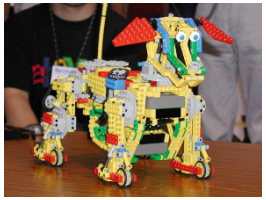
- Exponential growth in data size and rate (50 billion IoT By 2020)
- Demand for real time (near real time) data processing is becoming the norm
- Conventional servers cannot cope with the processing power and performance requirement

Cloud computing and datacenters

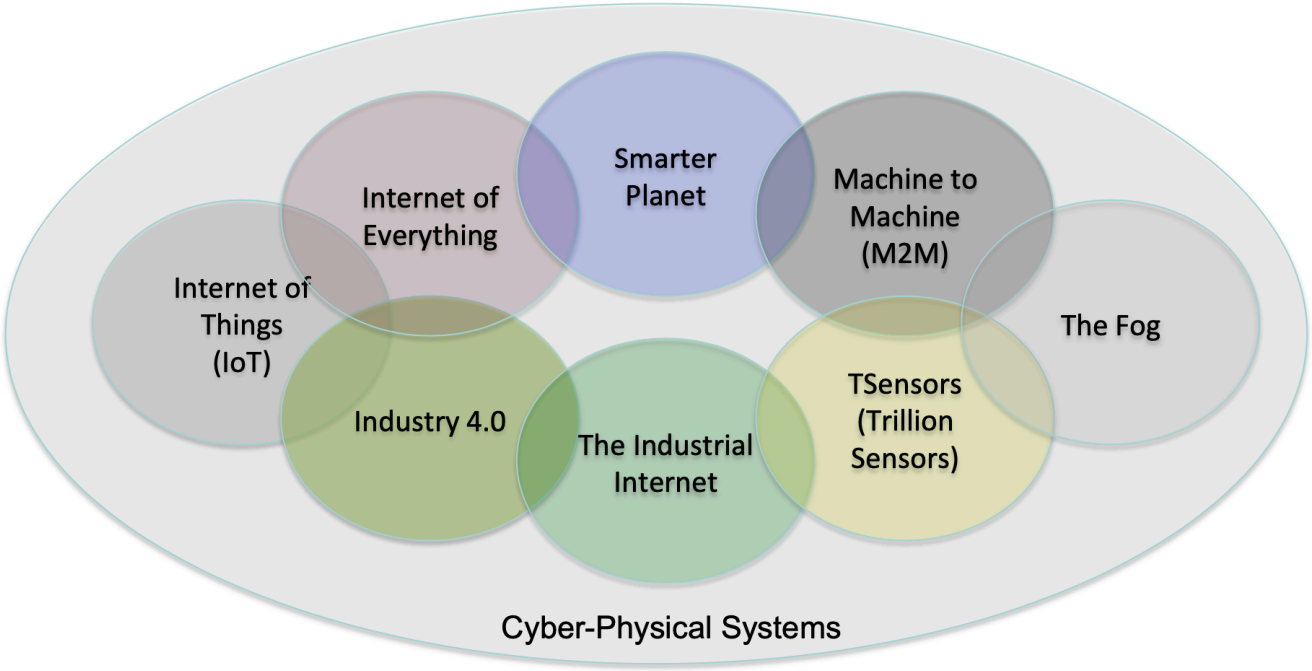
- Huge growth in the number of IoT devices
  - In 5 years more than 41.6 billion IoT devices
  - More than 79.4 zettabytes of data
- Advancements in Communication
  - 5G: up to 20 Gbps peak, and 100+ Mbps on average
- The cloud and edge computing

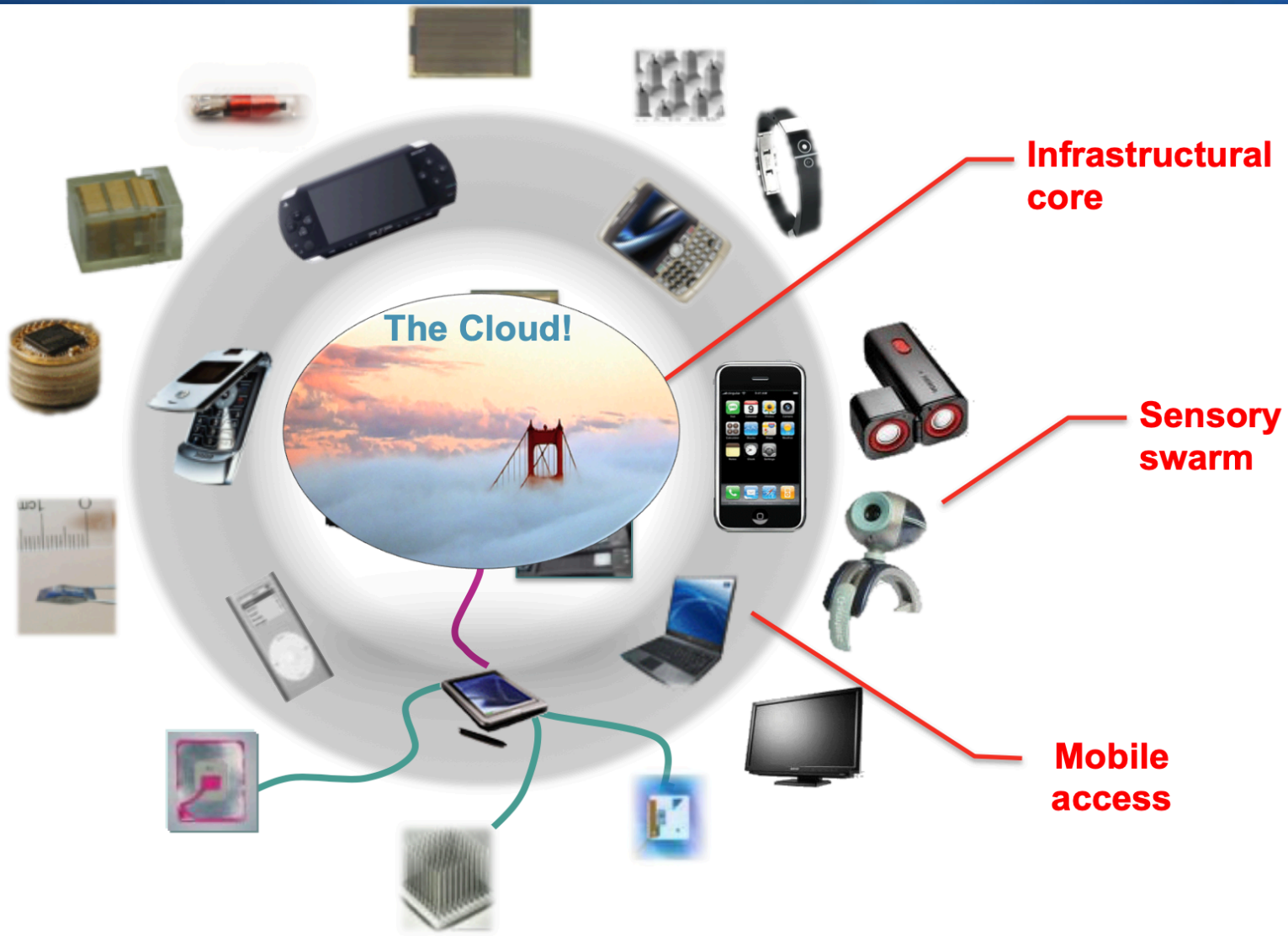
Motivation for course

- According to forecasts, future of IT characterized by terms such as
  - Disappearing computer,
  - Ubiquitous computing,
  - Pervasive computing,
  - Ambient intelligence,
  - Post-PC era,
  - Cyber-physical systems.
- Basic technologies:
  - *Embedded System technologies*
  - Communication technologies



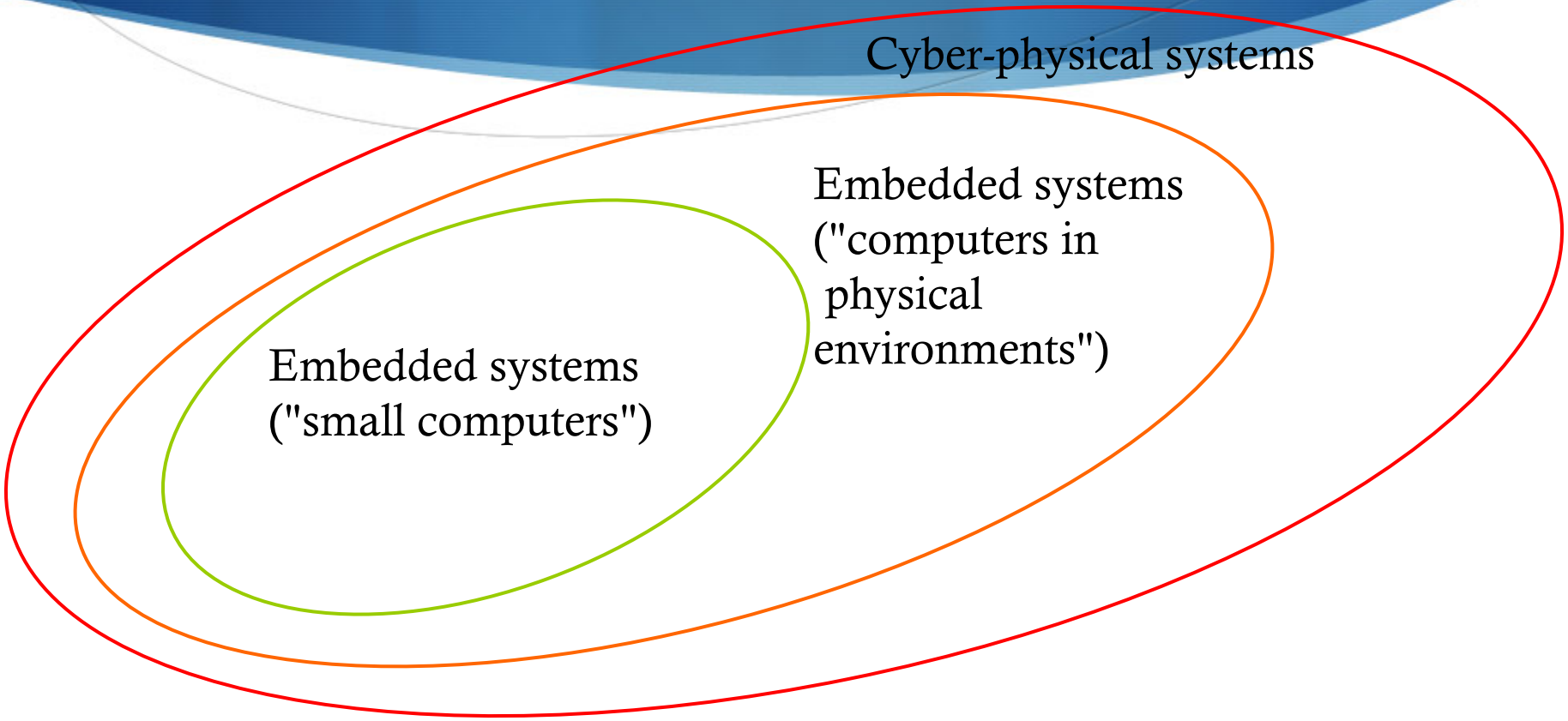
Cyber-Physical Systems





Courtesy: J. Rabaey

Cyber-physical systems and embedded systems



CPS = ES + physical environment

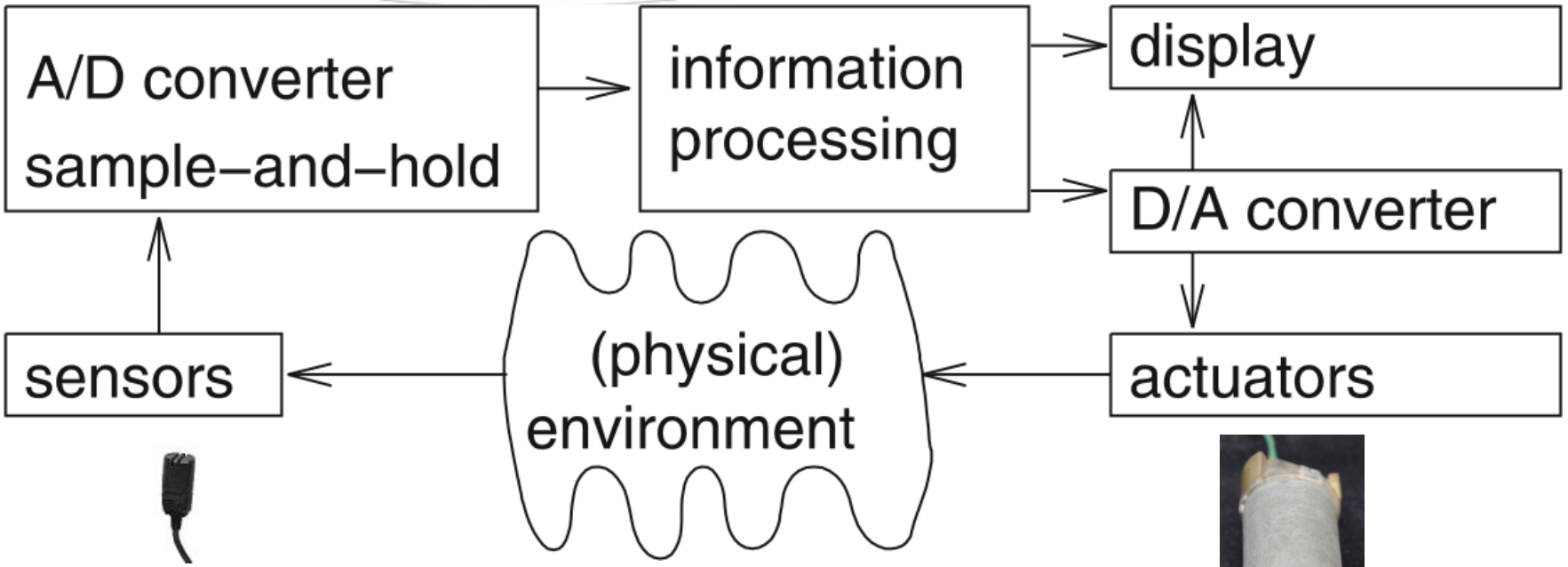
CPS: Integration of Cyber and Physics

Cyber

Physics

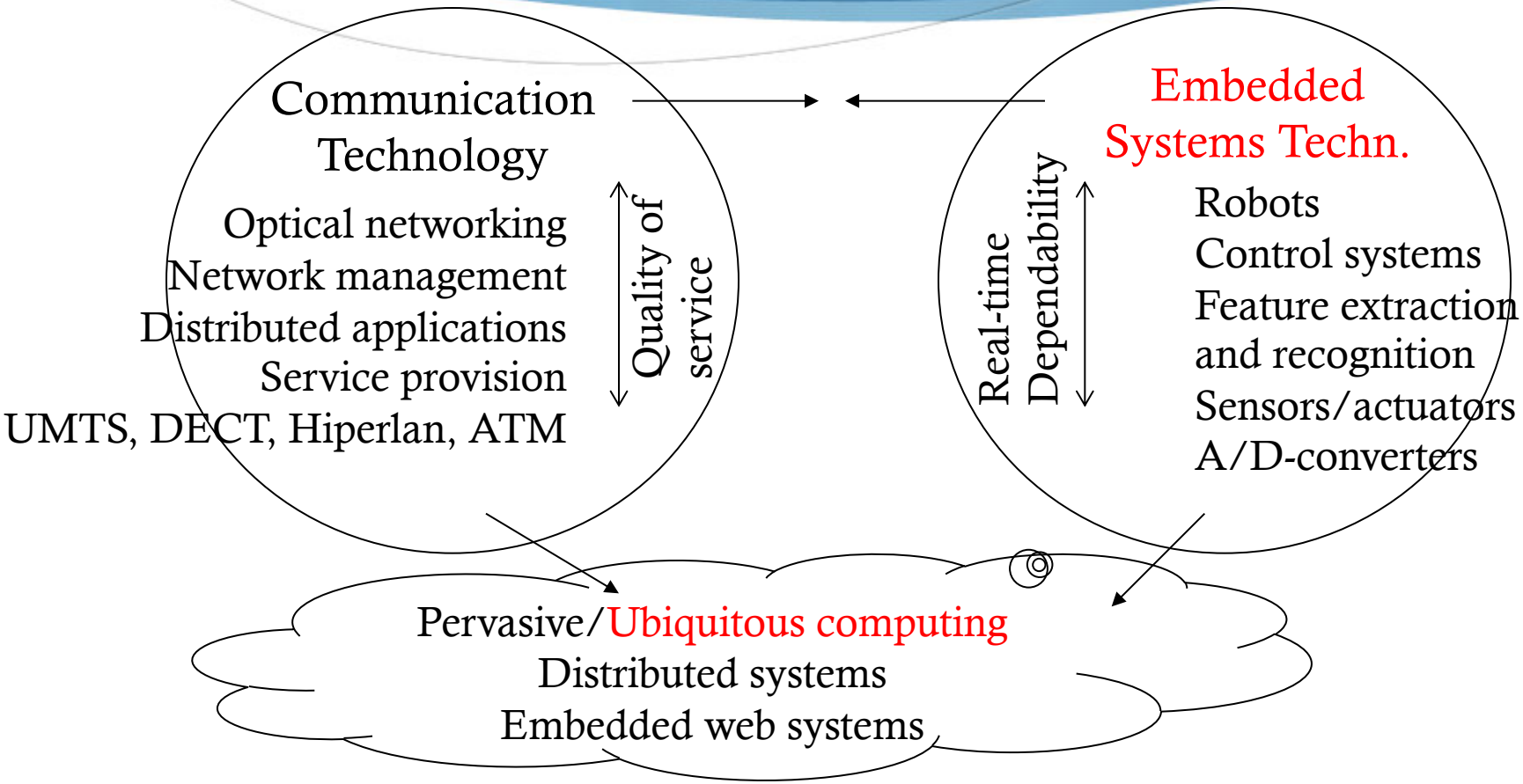
CPS

CPS & ES Hardware



Cyber-physical systems (!)

Extending the motivation: Embedded systems and ubiquitous computing



Application area automotive electronics: clearly cyber-physical

Functions by embedded processing:

- ABS: Anti-lock braking systems
- ESP: Electronic stability control
- Airbags
- Efficient automatic gearboxes
- Theft prevention with smart keys
- Blind-angle alert systems
- … etc …

- Multiple networks
- Multiple networked processors

[Based on slide by J.Engblom]

Application area avionics: also cyber-physical

- Flight control systems,
- anti-collision systems,
- pilot information systems,
- power supply system,
- flap control system,
- entertainment system,
- …

- Dependability is of outmost importance.

More application areas:

- railroad

- water ways

Dependability is of outmost importance.

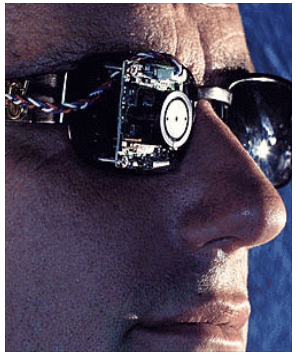
©Photos: P. Marwedel

Smart Home

- Zero energy building, generates as much energy as it consumes
- Provides safety and security
- Supports owners
- Provides maximum comfort
- ambient assisted living

Medical systems: cyber-physical

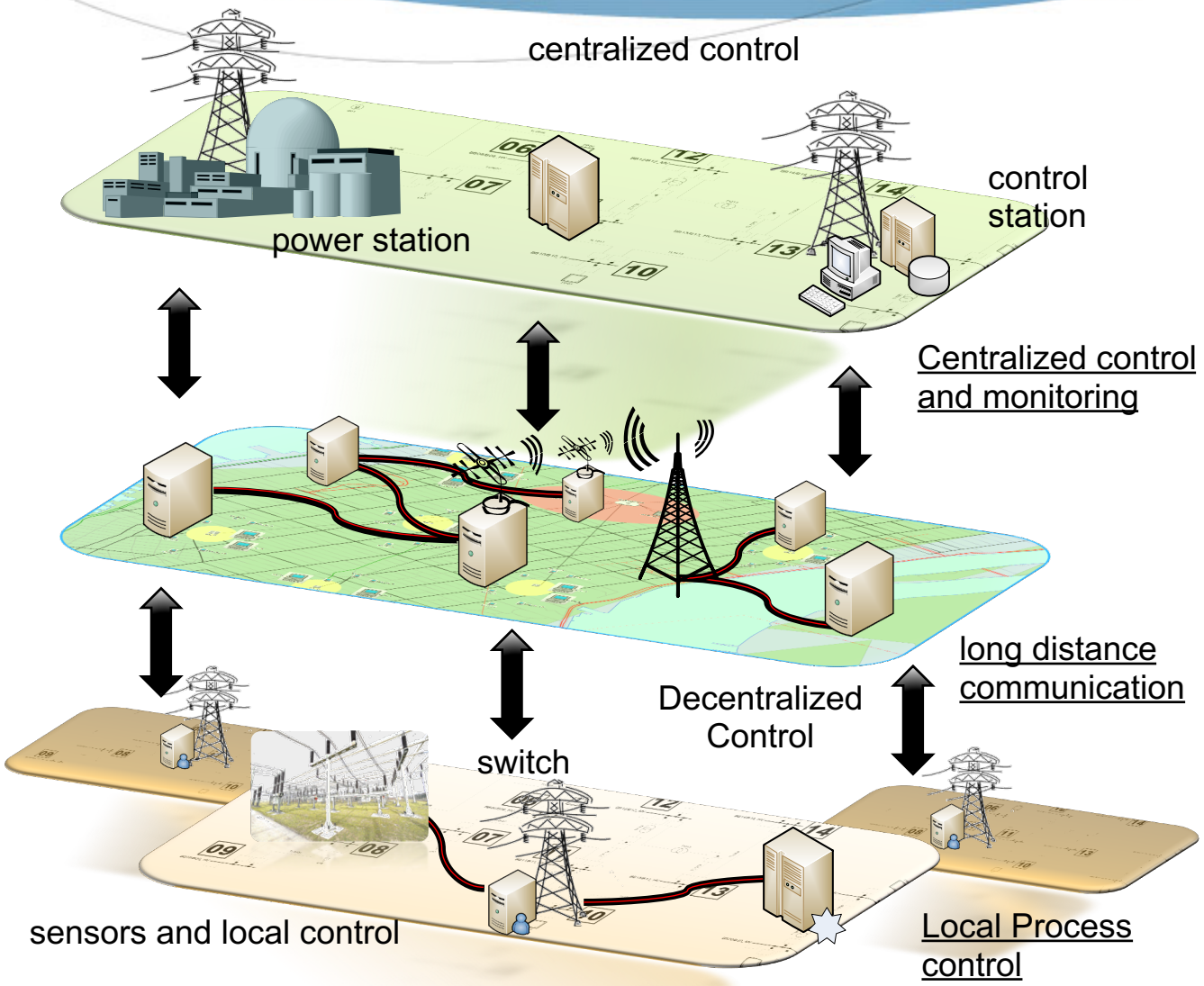
- For example:
  - Artificial eye: several approaches, e.g.:
    - Camera attached to glasses; computer worn at belt; output directly connected to the brain, "pioneering work by William Dobelle". Previously at [www.dobelle.com]



Smart Medicine

- Diagnosis
- Support of therapy
- evaluation
- risk analysis
- Information about patients

Smart Grids



Common characteristics

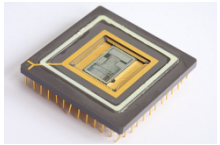
Dependability

- CPS/ES must be **dependable**,

  - **Reliability *R(t)* =** probability of system working correctly provided that is was working 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
  - **Security**: confidential and authentic communication

Efficiency

- CPS & ES must be **efficient**



  - Code-size efficient
    (especially for systems on a chip)

  - Run-time efficient

  - Weight efficient

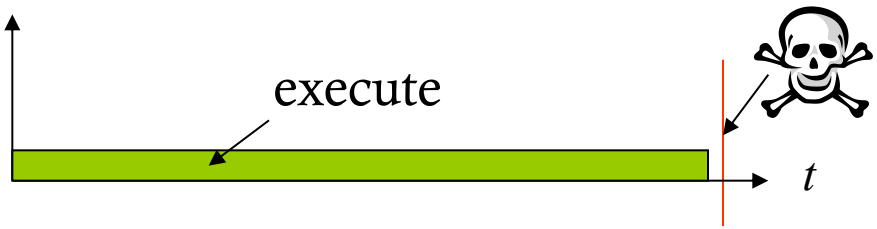
  - Cost efficient

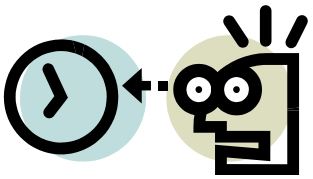
  - Energy efficient

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.
- CPS must meet **real-time constraints**

Real-time constraints





- **“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 [Kopetz, 1997].

Real-Time Systems & CPS

- CPS, ES and Real-Time Systems synonymous?
  - For some embedded systems, real-time behavior is less important (smart phones)
  - For some CPS, real-time behavior is essential, hence RTS ≅ CPS

Security

- Defending against
  - Cyber crime („Annual U.S. Cybercrime Costs Estima
    at $100 Billion; …[Wall Street Journal, 22.7.2013])
  - Cyber attacks (☞ Stuxnet)
- Connectivity increases threats
  - Entire production chains can be affected

Challenges for implementation in hardware

- Early embedded systems frequently implemented in hardware (boards)
- Mask cost for specialized application specific integrated circuits (ASICs) becomes very expensive (M$ range, technology-dependent)
- Lack of flexibility (changing standards).
- ☞Trend towards implementation in software (or possibly FPGAs)

Challenges for CPS/ES Software

- Dynamic environments
- Capture the required behaviour!
- Validate specifications
- Efficient translation of specifications into implementations!
- How can we check that we meet real-time constraints?
- How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)

© Graphics: P. Marwedel, 2011

Software complexity is a challenge

Software in a TV set

- Source 1*:

Year	Size
1965	0
1979	1 kB
1990	64 kB
2000	2 MB

- Source 2°: 10x per 6-7 years

Year	Size
1986	10 KB
1992	100 kB
1998	1 MB
2008	15 MB

- ☞ Exponential increase in software complexity
- *... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development* [A. Sangiovanni-Vincentelli, 1999]

* Rob van Ommering, COPA Tutorial, as cited by: Gerrit Müller: Opportunities and challenges in embedded systems, *Eindhoven Embedded Systems Institute*, 2004

° R. Kommeren, P. Parviainen: Philips experiences in global distributed software development, *Empir Software Eng*. (2007) 12:647-660

AI and Machine Learning on Embedded Devices

Embedded AI market

- Robotics
- Automotive and autonomous driving vehicles
- Medical
- Financial

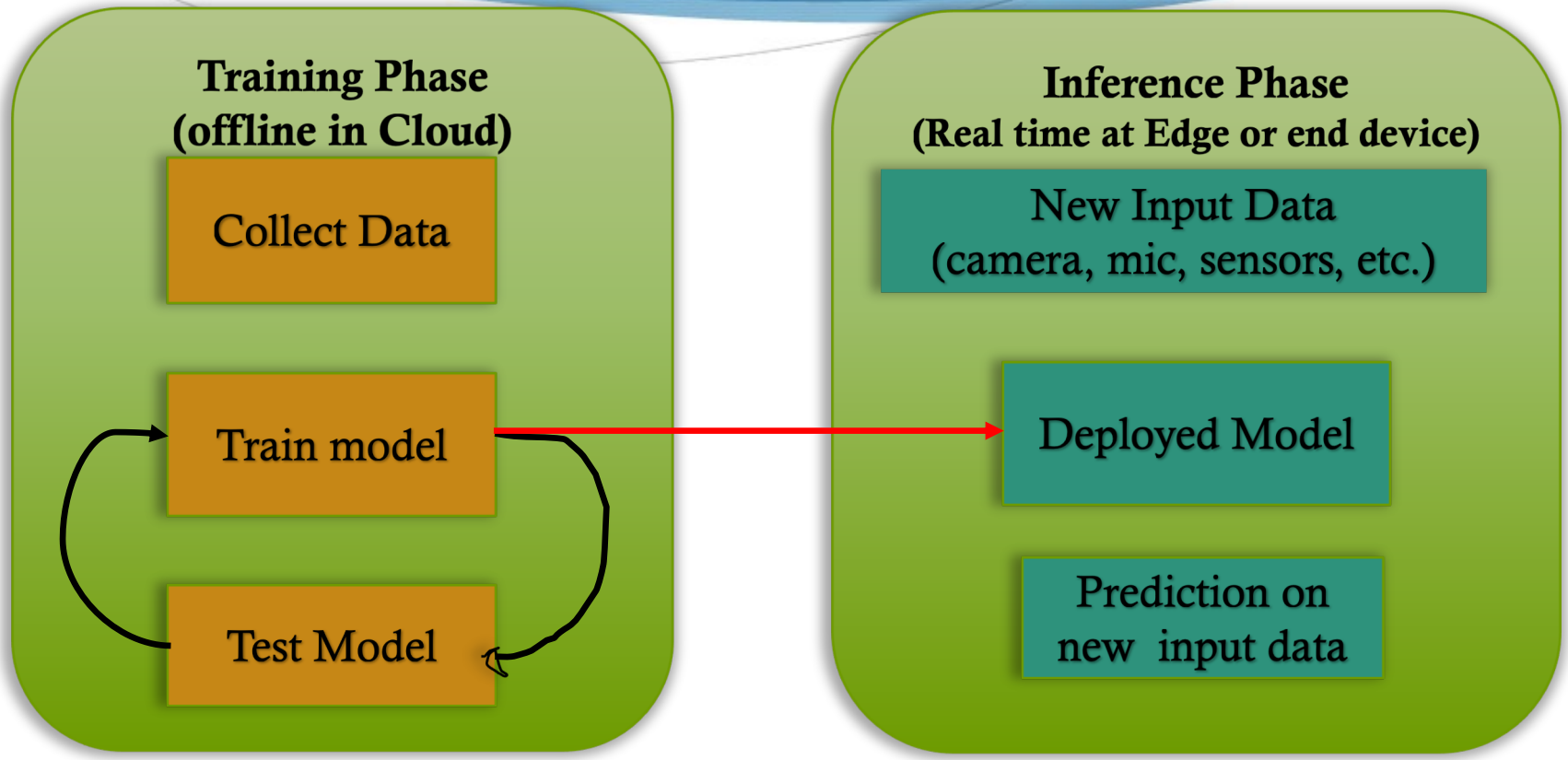
AI Embedded Applications

- Video Analytics
  - **Image Classifications :** Identify what Camera is looking at (Truck vs a passenger, coffee shop, factory defects, etc.)
- Audio Analytics
  - **Keyword actions:** "Alexa"
  - **Voice Commands**
  - **Alarm Analytics :** explosion, breaking glass, etc.
- Anomaly Detection
  - **Wearable human Health monitoring**
  - **Engine health monitoring**

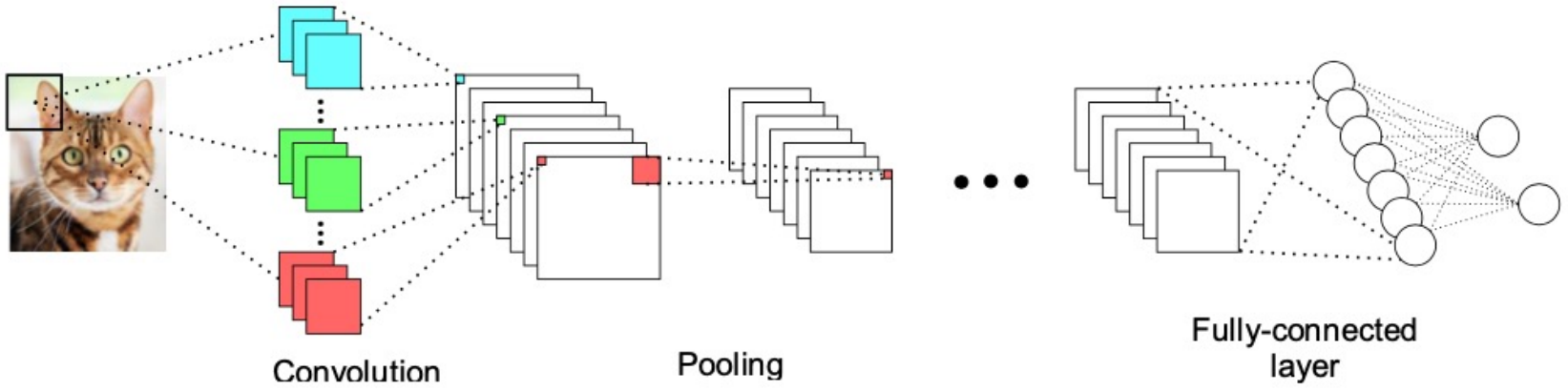
Embedded AI Technology

- AI model: MobileNet
- Data Input: Audio, Video, etc.
- AI Framework: TensorFlow, tensorFlow Lite, Café, MXNet
- Hardware:
  - Embedded processors: Raspberry pi
  - Accelerators: Google Coral AI Accelerator, Intel Neural Network Stick2, Invidia Jetson Nano, FPGAs, etc.

Machine learning process: Two phases



Structure of a Typical Deep Neural Network



Inference at Edge or End Device

- Faster (Avoids latency issues in cloud computing)
- Low power
- No network bandwidth issues
- Avoid cloud computing costs
- **Increased Privacy and Security**

Deep Learning on ARM

- Need compact model that fit within the ARM memory system
- Need models with fewer and simpler operations to achieve real time performance
- Neural Network model parameters:
  - NN architecture, Number of input features, Number of layers, types of layers, etc.
- High level languages like Python are used to build ML models whereas C/C++ like languages are used to program small embedded processors and MCU (Micro Controller Unit)

MCU Comparisons

Device Name	Clk Frequency	Flash	SRAM	Power
Arduino MKR1000 (Cortex-M0)	48 MHz	256KB	32KB	4mA
Arduino Due (Cortex-M3)	84 MHz	512 KB	96KB	50 mA
RPI3	1.2 GHz	SD Card	1 GB	260 mA

Raspbery pi is not as resource constrained as other MCUs in this table. RPIs can run high level languages like Python. Others may not.

How to run ML on small embedded devices?

- MCU + GPU technology
  - https://www.microchip.com/design-centers/32-bit/pic-32-bit-mcus/pic32mz-da-family
- Architectural Features for ML processing
  - Helium technology by ARM for Cortex-M family MCUs:
    - A new vector instruction set extension (MVE)
    - Additional instruction set enhancements for loops and branches (Low Overhead Branch Extension)
    - Instructions providing half precision floating-point support
    - Instruction improving Floating Point Unit (FPU)

How to run ML on small embedded devices?

- Sensors + ML core
  - STM ISM330DHCX :
    - https://www.st.com/en/mems-and-sensors/ism330dhcx.html
    - 3D digital accelerometer and a 3D digital gyroscope + ML core

Machine Learning algorithms for Embedded Systems

- ProtoNN: based on kNN
  - Uses entire training dataset during inference
  - Not suitable for memory-constrained embedded devices
- Bonsai: based on decision tree
- SeeDot: a framework to build ML models for devices without floating point units (using fixed-point data)
- ARM CMSIS-NN: ML library for Cortex-M processors
- TensorFLow Lite, FastGRNN, fastRNN, etc…

References

- CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs (https://arxiv.org/pdf/1801.06601.pdf)