Introduction to Embedded System

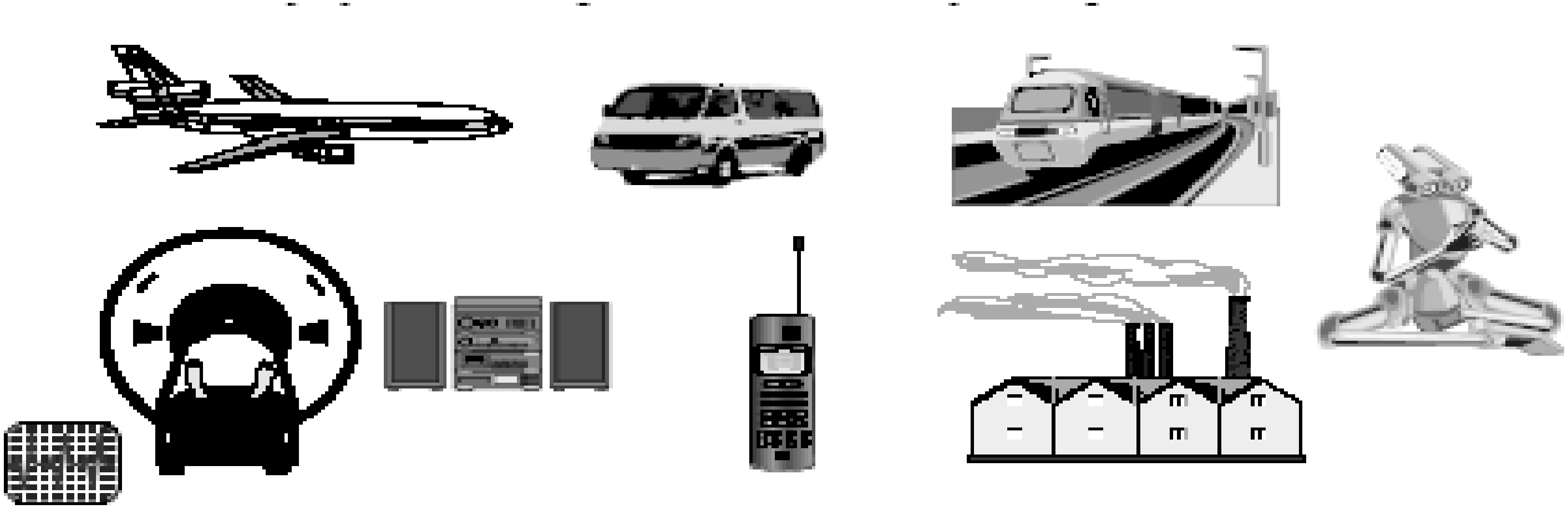
www.SarwanSingh.com

Embedded Systems: An Introduction

- What is an embedded system?
  - More than just a computer
- What makes embedded systems different?
  - Real-time operation
  - *Many* sets of constraints on designs
    - size
    - cost
    - time
    - reliability
    - safety
    - energy
    - security
- What embedded system designers need to know?
  - The "big" picture
  - Skills required to be an "expert" in this area

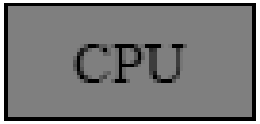
What is an Embedded System?

- Computer purchased as part of some *other* piece of equipment
  - – Typically dedicated software (may be user customizable)
  - – Often replaces previously electromechanical components
  - – Often no “real” keyboard
  - – Often limited display or no general purpose display device
- But, every system is unique there are always exceptions



CPU: An All-Too-Common View of Computing

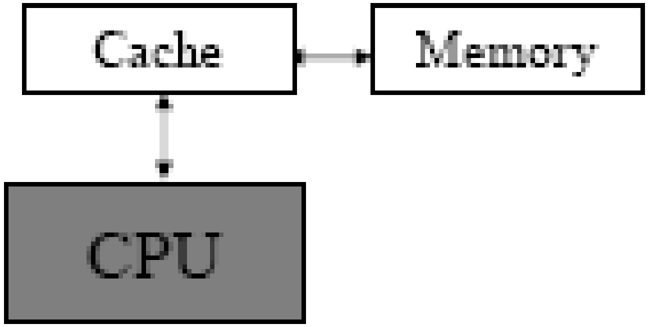
- Measured by:
  - Performance



CPU

An Advanced Computer Engineer's View

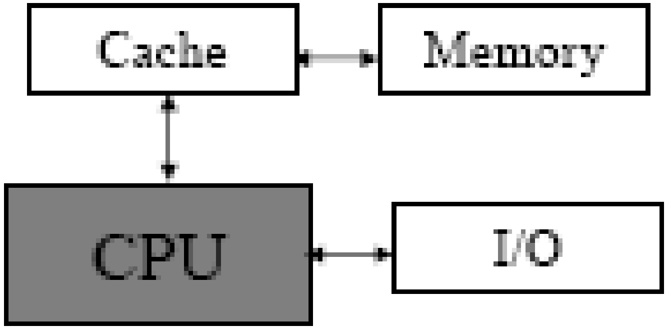
- Measured by: Performance
  - Compilers matter too...



An Enlightened Computer Engineer's View

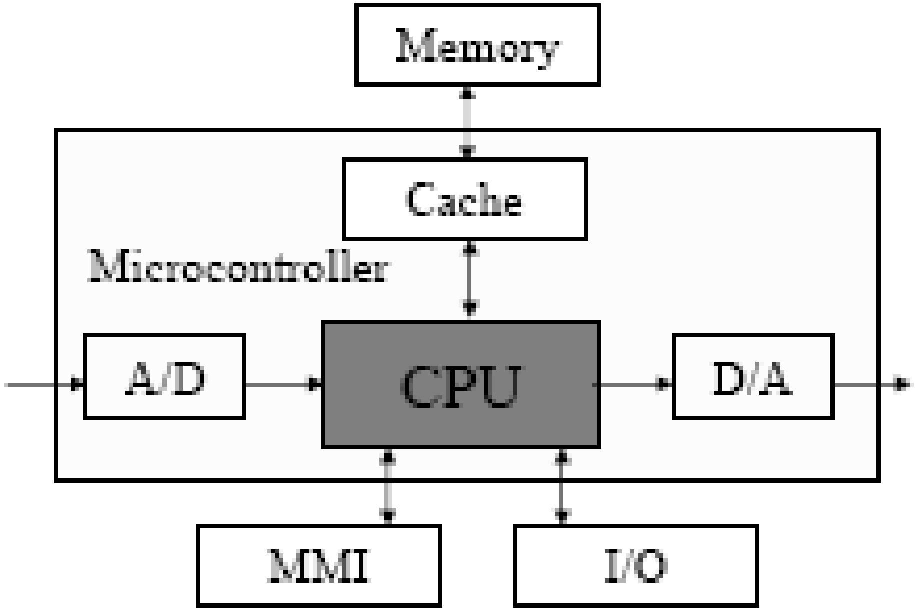
- Measured by: Performance, Cost

  Compilers & OS matters



An Embedded Computer Designer's View

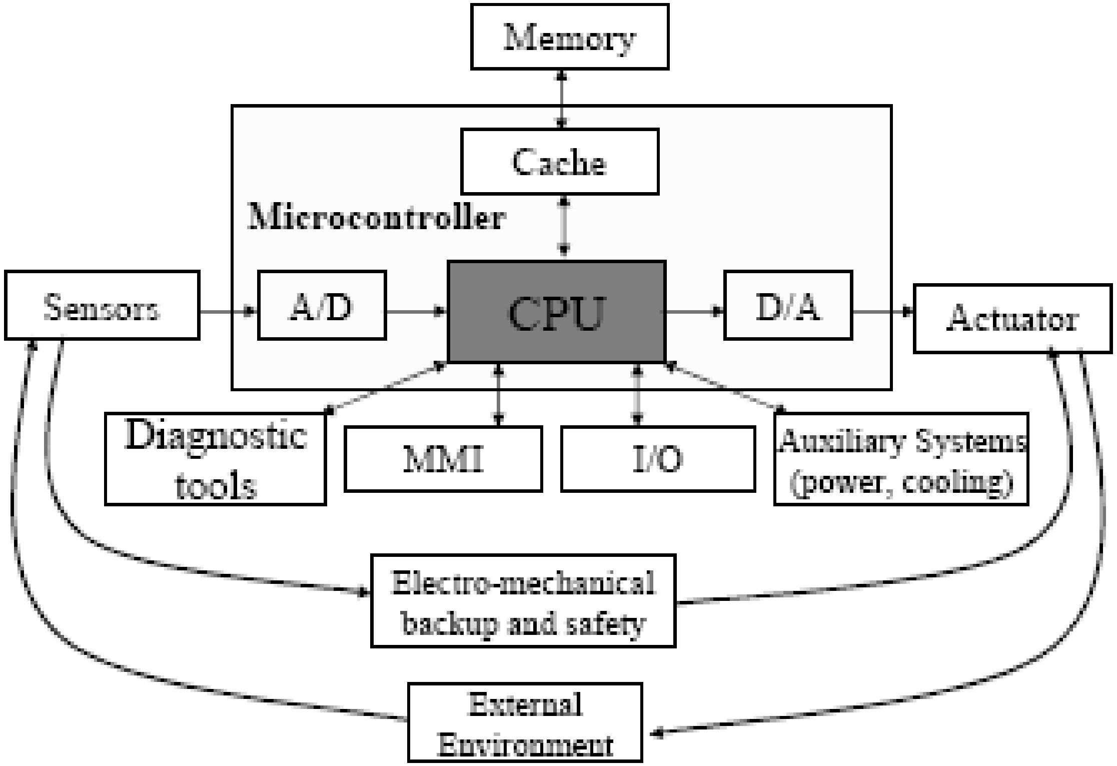
- Measured by: Cost, I/O connections, Memory Size, Performance



An Embedded Control System Designer's View

- Measured by:

  Cost, Time to market, Cost, Functionality, Cost & Cost.

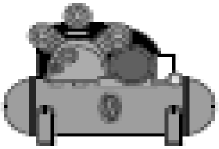


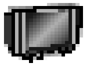
A Customer View

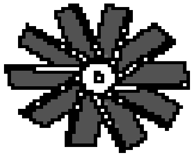
- Reduced Cost
- Increased Functionality
- Improved Performance
- Increased Overall Dependability

Some Embedded System Examples

- Pocket remote control RF transmitter
  - 100 KIPS, water/crushproof, fits in pocket, 5year battery life
  - Software handcrafted for small size (less than 1 KB)
- Industrial equipment controller (e.g., elevator; jet engine)
  - 110 MIPS for 1 to 10 CPUs, 1 8MB memory
  - Safety critical software; real time control loops
- Military signal processing (e.g., Radar/Sonar)
  - 1 GFLOPS, 1 GB/sec I/O, 32 MB memory
  - Software handcrafted for extremely high performance







Embedded Computers *Rule* the Marketplace

- ~80 Million PCs vs. ~3 Billion Embedded CPUs annually

– Embedded market growing; PC market *mostly* saturated

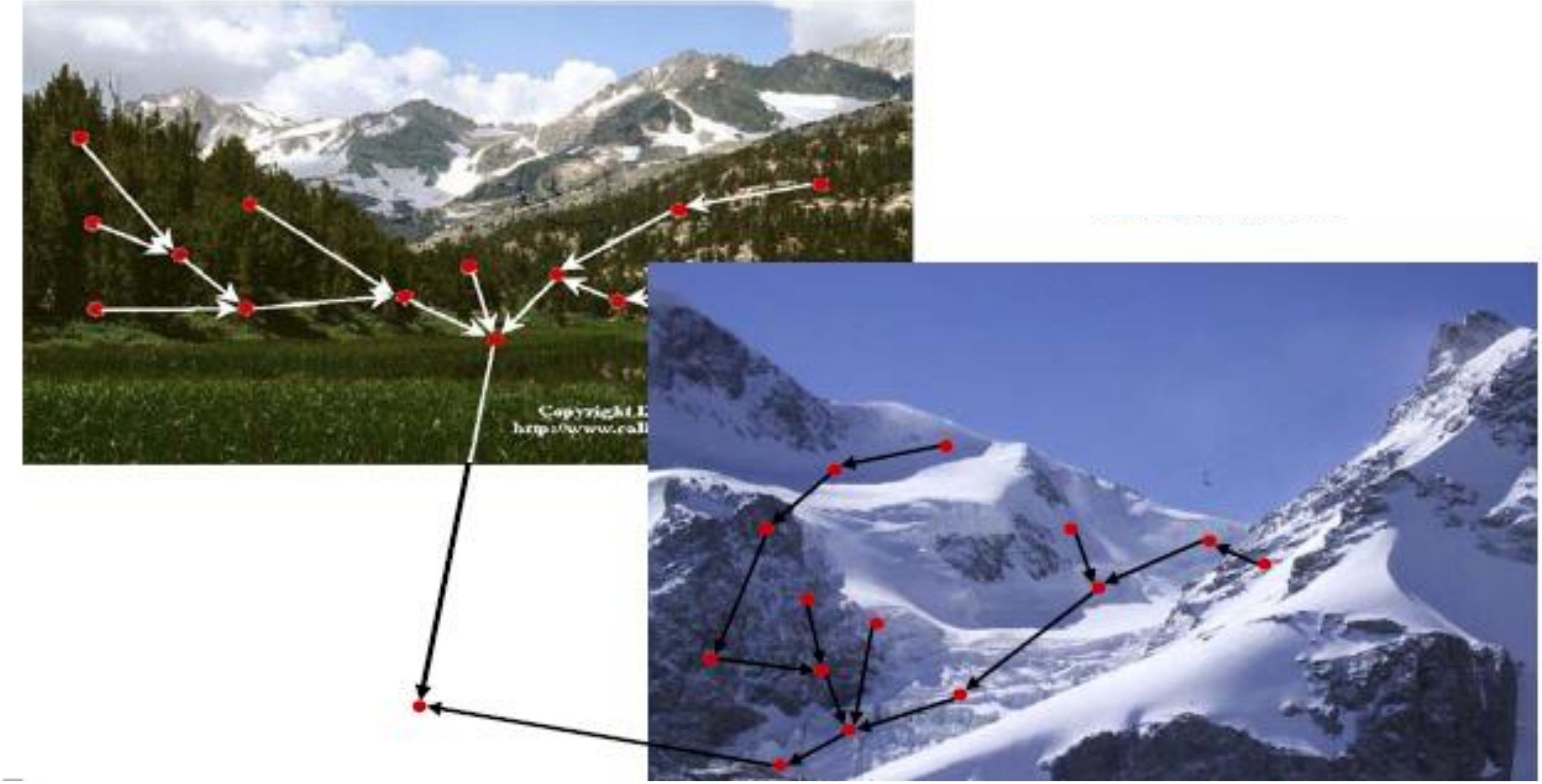
Why Are Embedded Systems Different?

Four General Categories of Embedded Systems

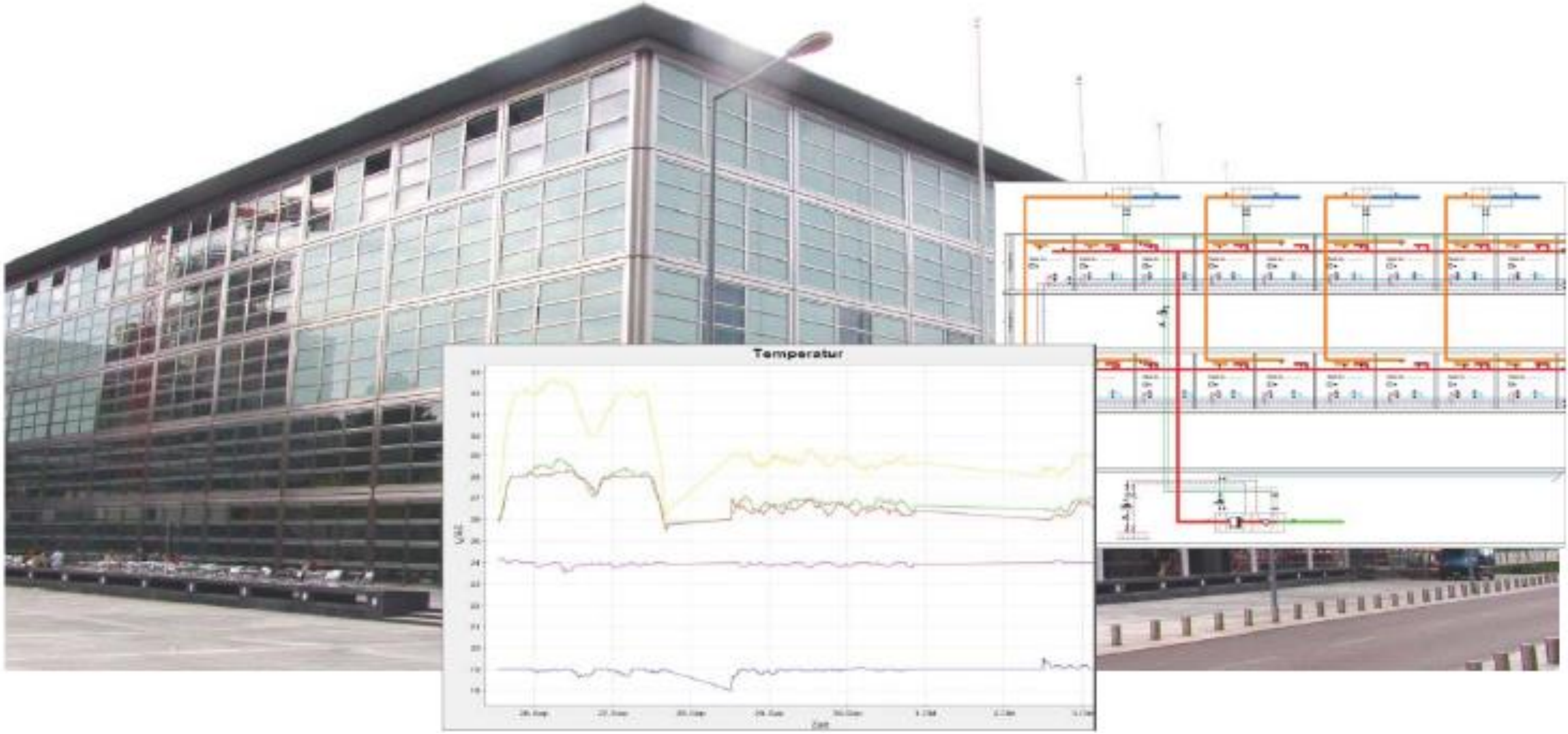
- General Computing
  - – Applications *similar* to desktop computing, but in an embedded package
  - – Video games, set top boxes, wearable computers, automatic tellers
- Control Systems
  - – Closed loop feedback control of real time system
  - – Vehicle engines, chemical processes, nuclear power, flight control
- Signal Processing
  - – Computations involving large data streams
  - – Radar, Sonar, video compression
- Communication & Networking
  - – Switching and information transmission
  - – Telephone system, Internet

Communicating Embedded Systems

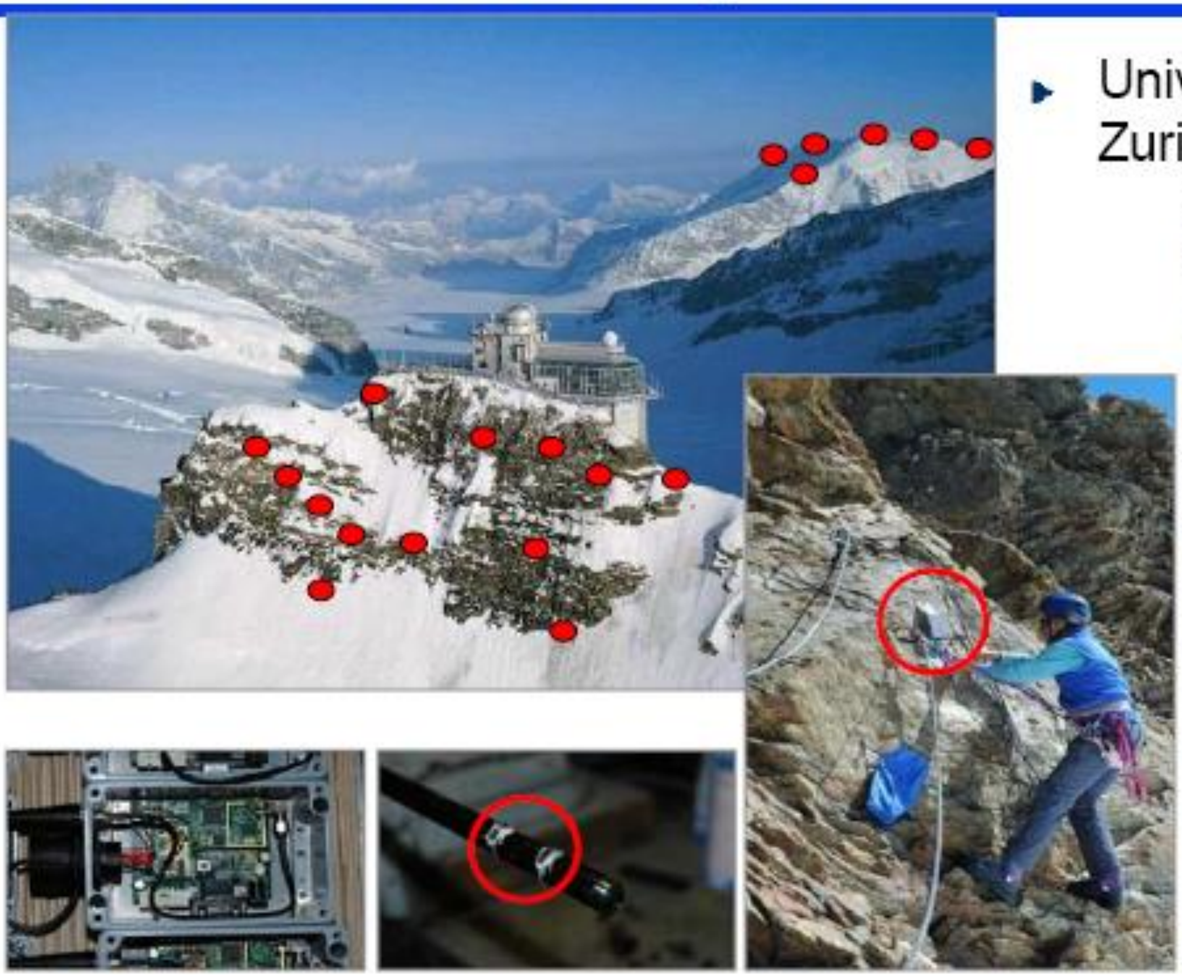
- sensor networks (civil engineering, buildings, environmental monitoring, traffic, emergency situations)
- smart products, wearable/ubiquitous computing



Communicating Embedded Systems

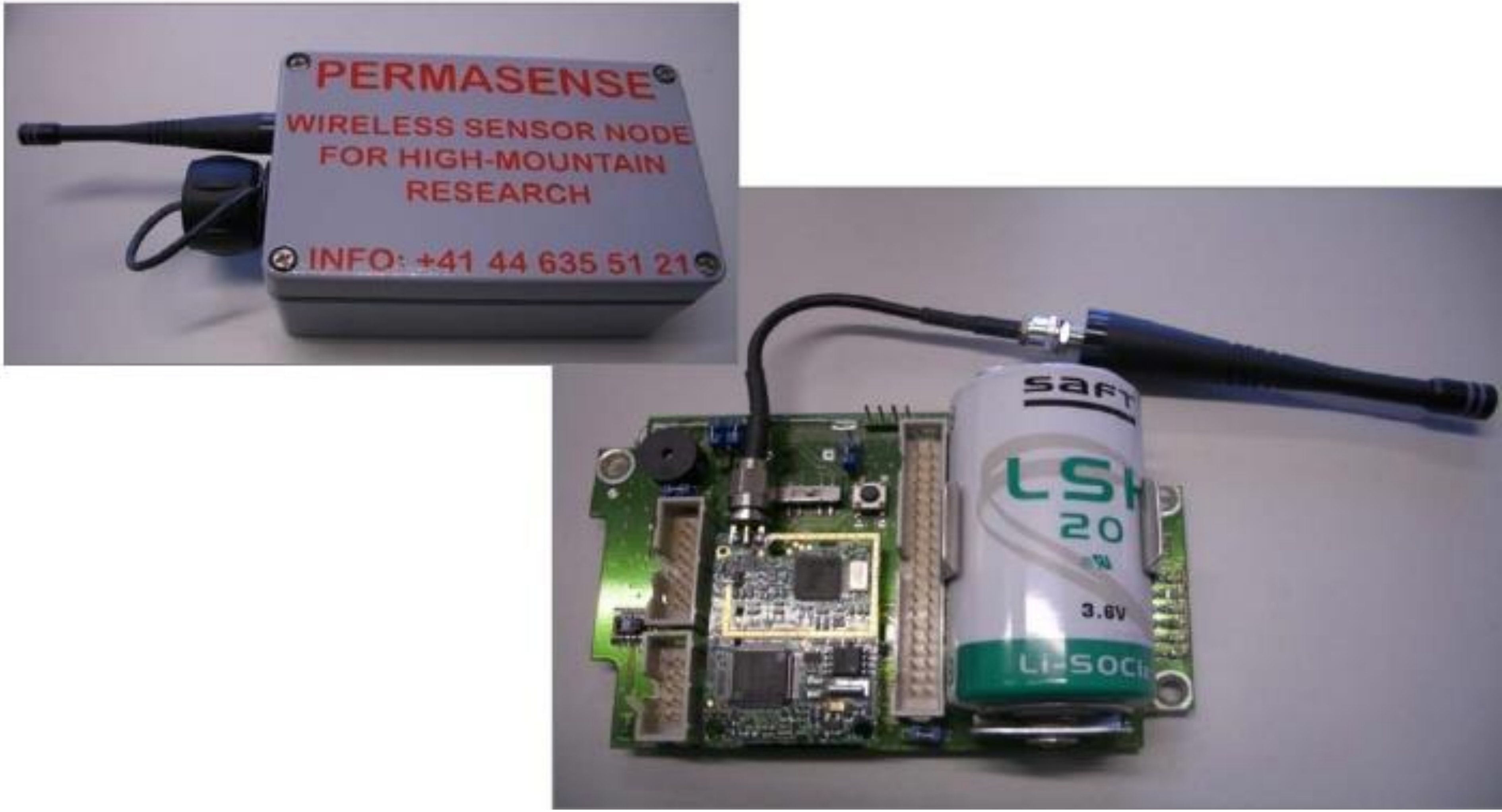


PermaSense Project

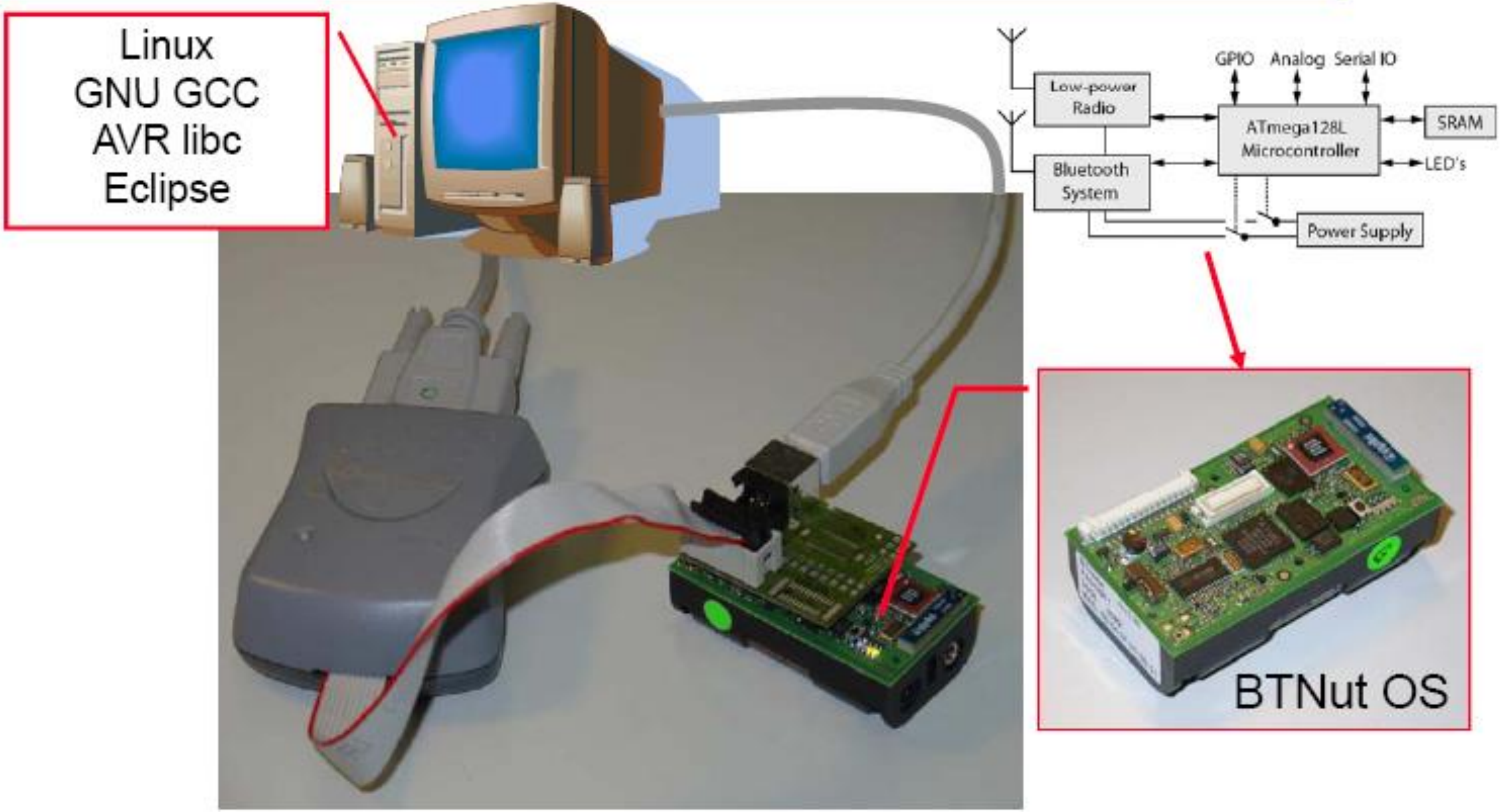


- Univ. Zurich, Univ. Basel, ETH Zurich

Hardware


PERMASENSE
WIRELESS SENSOR NODE
FOR HIGH-MOUNTAIN
RESEARCH
INFO: +41 44 635 51 21
saft
LSH
20
3.6V
Li-SOCl

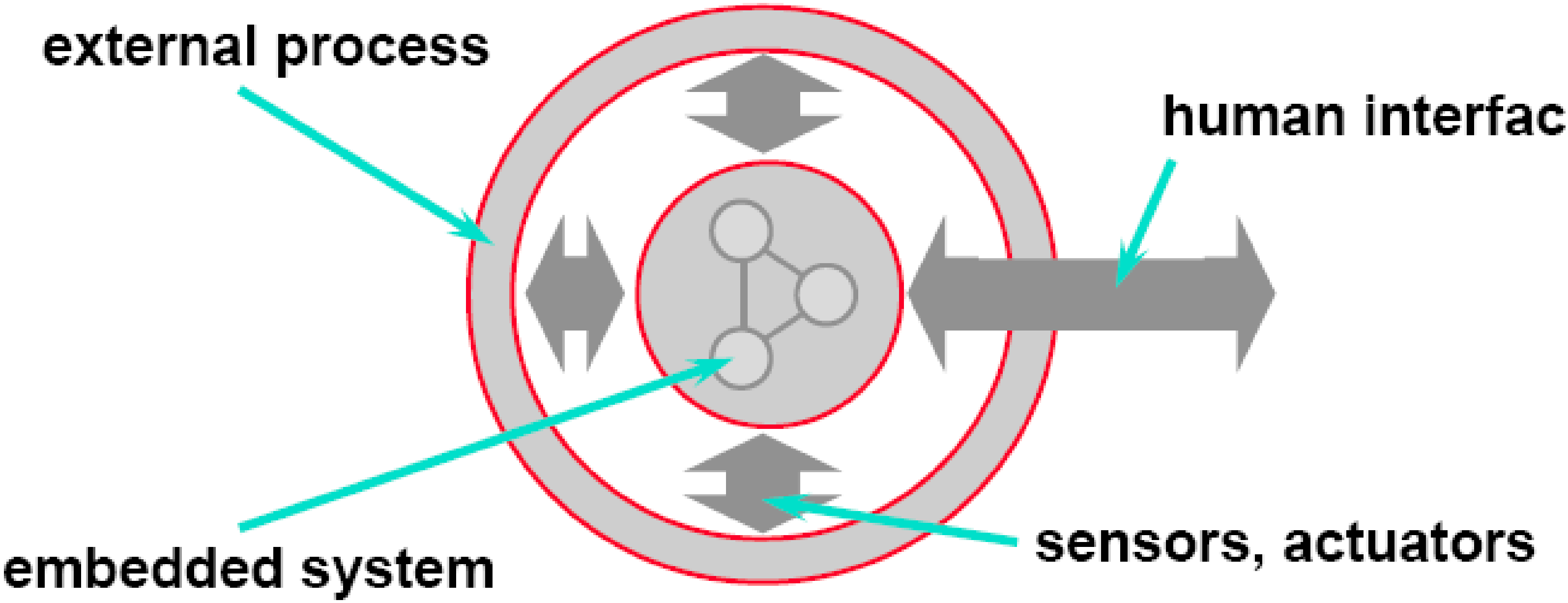
Development in ES Exercise



Linux
GNU GCC
AVR libc
Eclipse

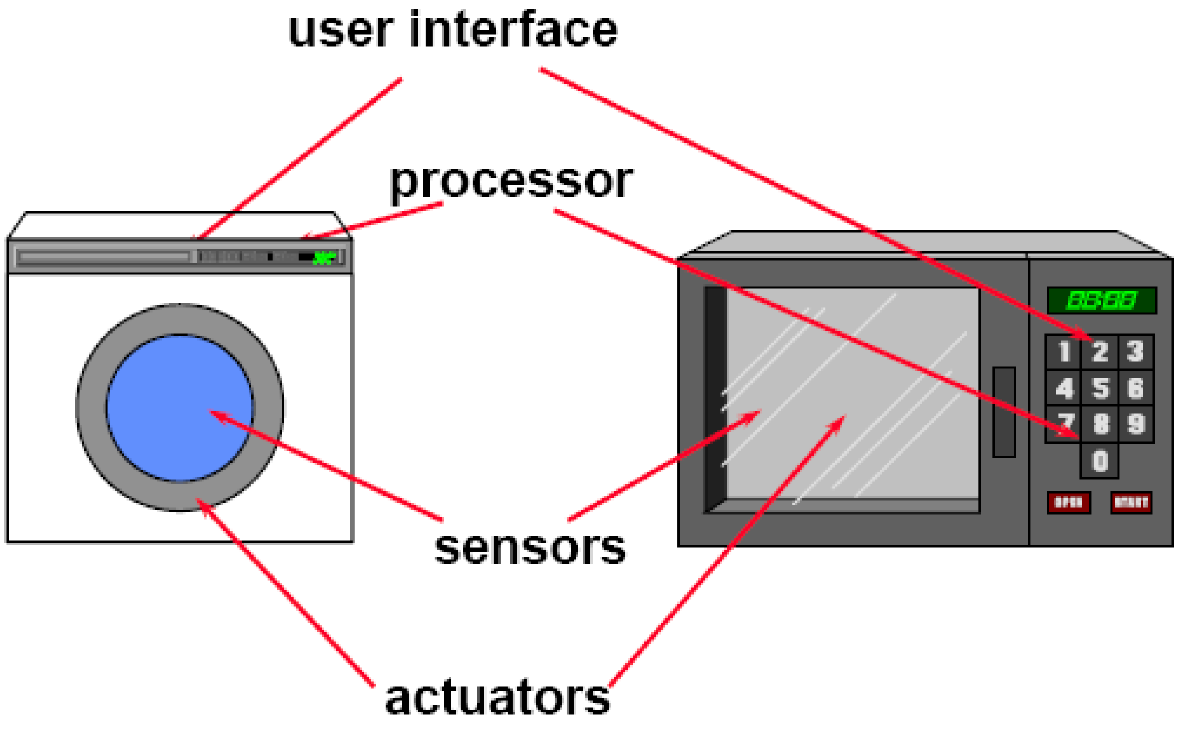
BTNut OS

Embedded Systems



Examples of Embedded Systems

Consumer electronics, for example MP3 Audio, digital camera, home electronics, … .



user interface

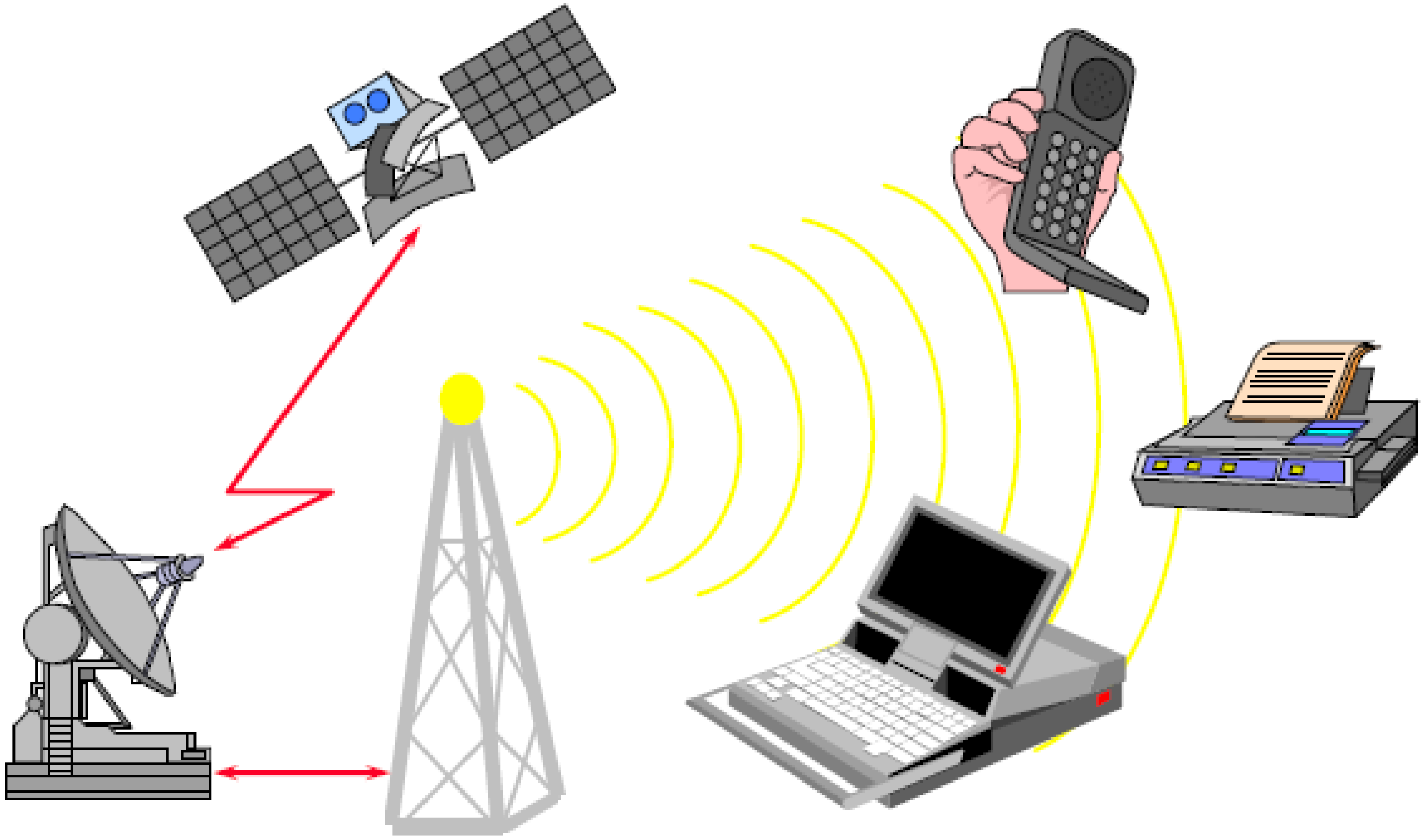
processor

sensors

actuators

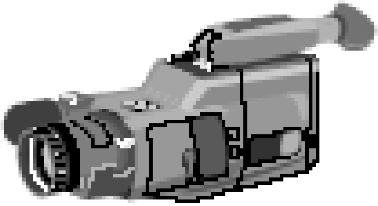
Examples of Embedded Systems

Information systems, for example wireless communication (mobile phone, Wireless LAN, …), end-user equipment, router, …



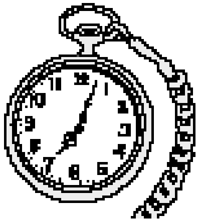
Types of Embedded System Functions

- Control Laws
  - – PID control
  - – Fuzzy logic, ...
- Sequencing logic
  - – Finite state machines
  - – Switching modes between control law
- Signal processing
  - – Multimedia data compression
  - – Digital filtering
- Application specific interfacing
  - – Buttons, bells, lights,...
  - – High speed I/O
- Fault response
  - – Detection & reconfiguration
  - – Diagnosis
- ...



Distinctive Embedded System Attributes

- **Reactive**: computations occur in response to external events
  - Periodic events (e.g., rotating machinery and control loops)
  - Aperiodic events (e.g., button closures)
- **Real-Time: timing correctness is part of system correctness**



– Hard real-time

- Absolute deadline, beyond which answer is useless
- May include minimum time as well as maximum time

– Soft real-time

- Missing a deadline is not catastrophic
- Utility of answer degrades with time difference from deadline

– ***Example***:

- a train is entering an urban area...
- the railway gate in the city allows automotive traffic to go over the tracks
- when should the railway gate close?

In general,

Real Time != “Real Fast”

The Patriot Missile Failure

- http://www-users.math.umn.edu/~arnold/disasters/patriot.html

Typical Embedded System Constraints

- Small Size, Low Weight
  - Handheld electronics
  - Transportation applications weight costs money
- Low Power
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- Harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse

- Safety critical operation
  - Must function correctly
  - Must not function incorrectly

- Extreme cost sensitivity
  - $.05 adds up over 1,000,000 units

Embedded System Design World-View

A complex set of tradeoffs:

- Optimize for *more than just speed*
- Consider *more than just the computer*
- Take into account *more than just initial product design*

Multi-Discipline

- Electronic Hardware
- Software
- Mechanical Hardware
- Control Algorithms
- Humans
- Society/Institutions

x

MultiPhase

- Requirements
- Design
- Manufacturing
- Deployment
- Logistics
- Retirement

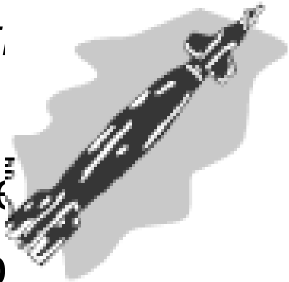
x

MultiObjective

- Dependability
- Affordability
- Safety
- Security
- Scalability
- Timeliness

Mission Critical Applications Require Robustness

- **Loss of Arianne inaugural flight in June, 1996**
  - – Lost a $400 million scientific payload (the rocket was *ext*



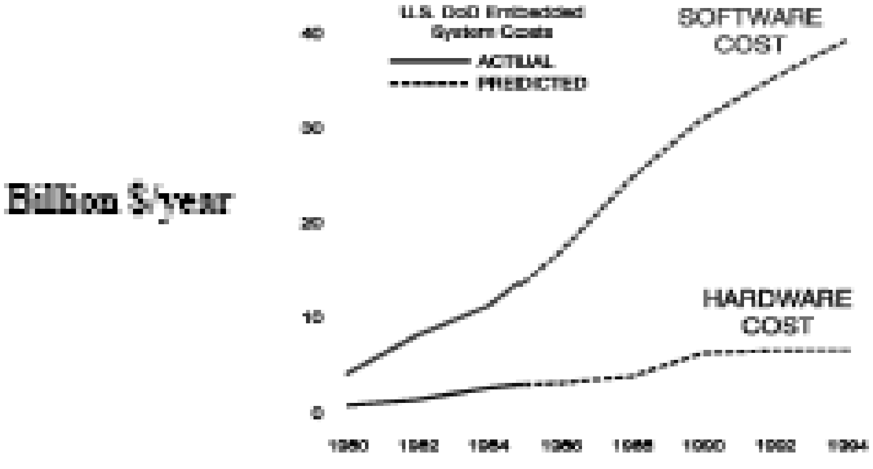
- **Efforts to reduce system costs led to the failure**
  - – Reuse of Inertial Reference System software from Arian
  - – Improperly handled exception caused by variable overflo
  - – new flight profile (that wasn't simulated because of cost/schedule)
  - – 64bit float converted to 16bit int assumed not to overflow
- Exception caused dual hardware shutdown (software doesn't fail!)
- **What really happened?**
  - – The narrow view: it was a software bug fix it
  - – The broad view: the loss was caused by a lack of system robustness in an

    exceptional (unanticipated) situation

Many embedded systems must be robust

Software Drives Designs

- Hardware is mostly a recurring cost
  - – Cost proportional to number of units manufactured
- Software is a “one time” nonrecurring engineering design cost (NRE)
  - – Paid for ``only once''
- But bug fixes may be expensive, or impossible
  - – Cost is related to complexity & number of functions
  - – Market pressures lead to feature creep
- **Software Is NOT free!!!!!**



Source: Software Requirements: objects, functions, states, Davis, 1993.

Life Cycle Concerns Figure Prominently

- “Let's use a CAD system to re-synthesize designs for cost optimization”
  - Automatically use whatever components are cheap that month
  - Would permit quick responses to bids for new variants
  - Track record of working fine for PC motherboards
- Why wouldn't it work for an automotive application?
  - Embedded systems had more analog than digital mostly digital synthesis tool
  - Cost of recertification for safety, FCC, warrantee repair rate
  - Design optimized for running power, not idle power
- Car batteries must last a month in a parking lot
  - Parts cost didn't take into account lifecycle concerns
- Price breaks for large quantities
- Inventory, spares, end of life buy costs
  - Tool didn't put designs on a single sheet of paper
- Archive system paper-based -- how else do you read
  - 20 year old files?

Embedded System Designer Skill Set

- **Appreciation for multidisciplinary nature of design**
  - Both hardware & software skills
  - Understanding of engineering beyond digital logic
  - Ability to take a project from specification through production
- **Communication & teamwork skills**
  - Work with other disciplines, manufacturing, marketing
  - Work with customers to understand the real problem being solved
  - Make a good presentation; even better write ``trade rag'' articles
- **And, by the way, technical skills too...**
  - Low-level: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
  - High-level: Object oriented Design, C/C++, Real Time Operating Systems
  - Meta-level: Creative solutions to highly constrained problems
  - Likely in the future: **U**nified **M**odeling **L**anguage, embedded networks
  - (Un)certain future: Java, Windows CE