COEN-4720 Embedded Systems Design
Lecture 1
Introduction

Cristinel Ababei
Dept. of Electrical and Computer Engineering
Marquette University

Outline

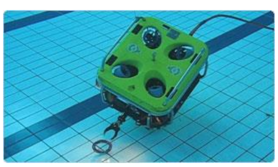
- What is an Embedded System (ES)
- Examples of embedded systems
- Embedded systems characteristics
- How to design an embedded system
- ARM Cortex-M3

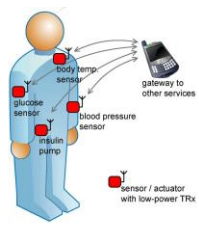
What is an Embedded System?

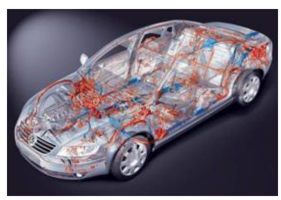
- Any electronic system that uses a computer chip, but that is not a general-purpose workstation, desktop or laptop computer
- An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular function
- An embedded system is a multi-agent system and computer system designed for specific control functions within a larger system, often with real-time computing constraints
- Many other definitions…

Embedded Systems

- Systems that are part of a larger system
  - Application-specific
    - Diverse application areas
- Tight constraints
  - Real-time, performance, power, size
  - Cost, time-to-market, reliability
- Ubiquitous
  - Far bigger market than general purpose computing (PCs, servers)
    - $46 billion in ‘04, >$90 billion by 2010, 14% annual growth
    - 4 billion devices in ‘04
    - 98% of all processors sold







Where are Embedded Systems Used?

- Everywhere
  - industrial machines
  - automobiles, trains
  - airplanes, space vehicles
  - medical equipment
  - video games, cameras, MP3 players, TVs
  - cell phones
  - vending machines, household appliances, toys
  - etc.

5

5

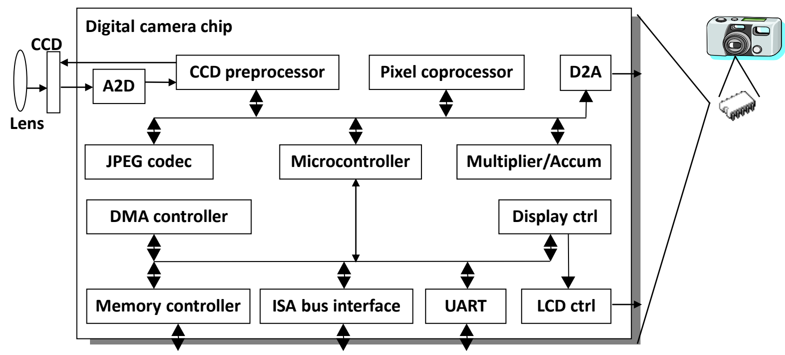
General Types of Embedded Systems

- General
  - similar to traditional computer systems, in a smaller package
  - PDA's
  - portable games
- Communications
  - cell phones
- Signal Processing
  - video and audio
- Control
  - real time feedback control
  - automotive
  - aerospace
  - appliances

6

6

Example of Embedded System: **Digital Camera**

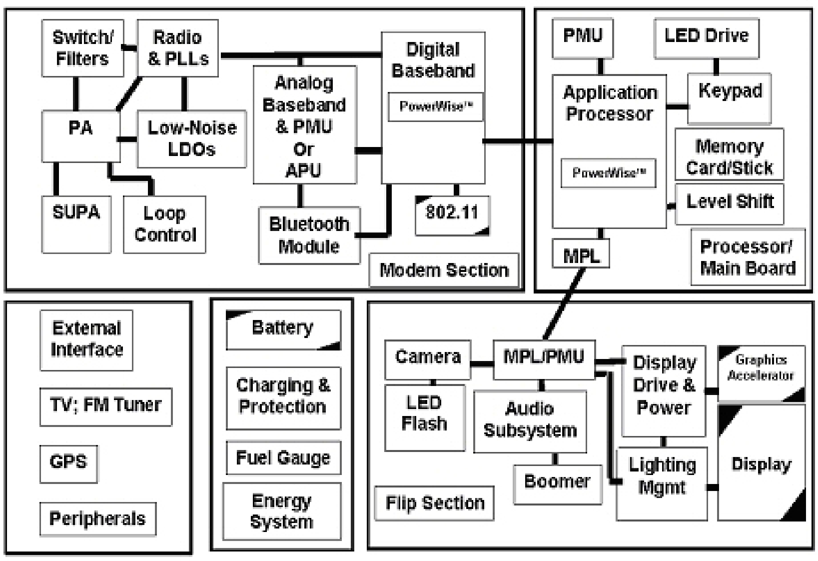


- Single functionality - always a digital camera
- Tightly constrained - low cost, low power, small, fast
- Reactive and real time - only to a small extent

7

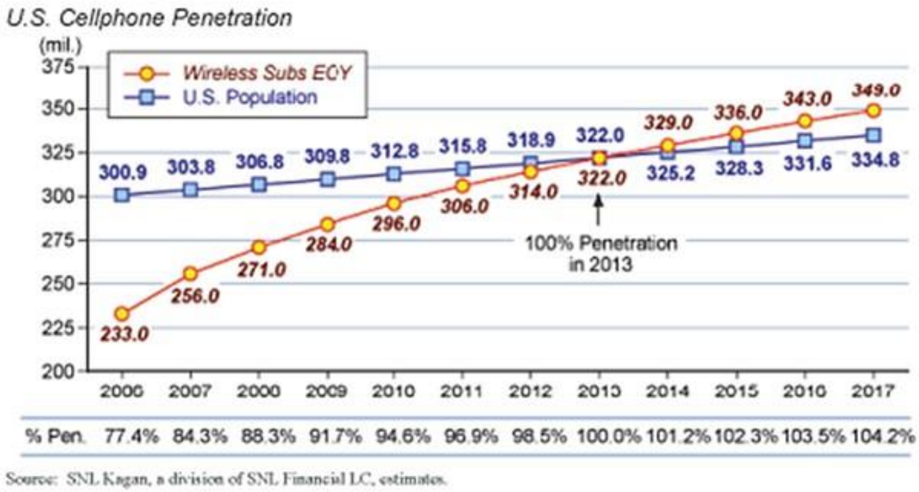
7

Example of Embedded System: **Mobile Phone**



8

Mobile phones: the most successful technology ever?



9

9

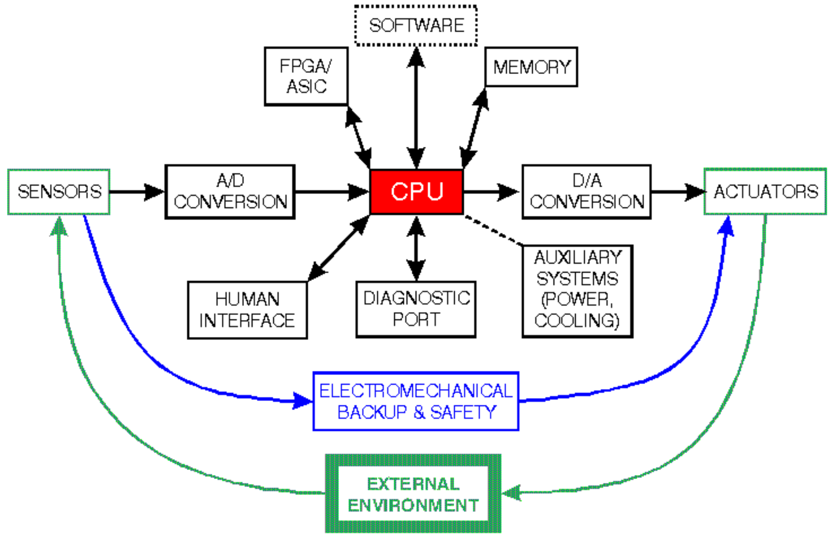
Outline

- What is an Embedded System (ES)
- Examples of embedded systems
- Embedded systems characteristics
- How to design an embedded system
- ARM Cortex-M3

10

10

Embedded Systems Characteristics



11

11

Embedded Systems Characteristics

- Part of a larger system (system within system)
- Computational
- Interact (sense, manipulate, communicate) with the external world: sensors, actuators
- Reactive: at the speed of the environment
- Heterogeneity: hardware/software blocks, mixed architectures
- Networked: shared, adaptive, sensor networks (buildings, environmental monitoring), smart products, wearable computing
- Flexibility: can run/implement multiple applications sequentially or concurrently - concurrency
- Reprogrammability/reconfigurability: flexibility in upgrading, bug fixing, product differentiation, product customization
- Performance and constraints:
  - Timing (frequency, latency, throughput)
  - Power consumption, area, temperature
  - Weight, size, cost (hardware & software), time to market
  - Real time critical, safety, reliability

12

12

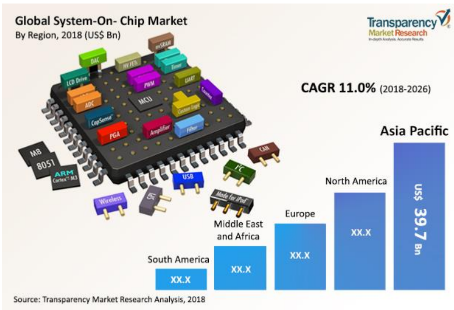
Key Recent Trends

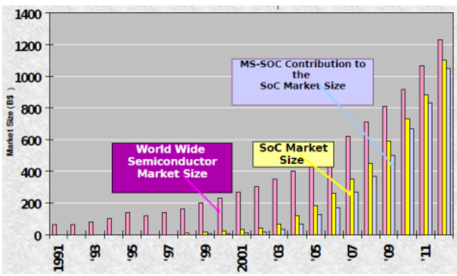
- Difficult to design
  - Planes still crash
  - Car recalls…
- Getting even harder to design:
  - Increasing computation demands, increasing complexity
    - e.g. multimedia processing in set-top boxes, HDTV
  - Increasingly networked and distributed
  - Increasing need for flexibility
    - programmable & customizable
    - time-to-market under ever changing standards
  - Reaching physical limits of technology scaling
    - Power walls (and dark silicon)
    - Efficiency/optimality vs. flexibility/generality

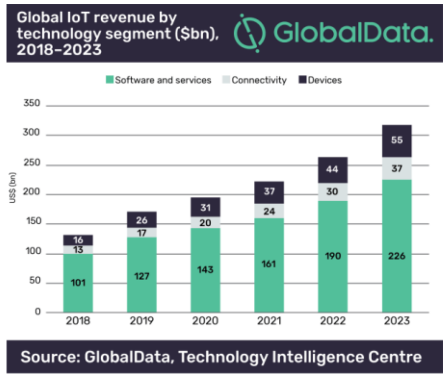
Key Recent Trends

- Technological advances
  - Higher integration: more blocks on the same chip
  - Multi-Processor System-On-Chip (MPSoC)
- Embedded systems evolve toward
  - System-on-Chip (SoC)
  - Cyber Physical Systems (CPS)
- IP reuse, platform based design, NoC vs. Bus
- HW-SW co-design
- Diversity in design methodologies, platform dependent, lack of standards, quality risks, customer confusion
- Systems are designed and built as “systems of systems”
- Opportunity and need for specialization
  - Heterogeneous multi-core / Asynchronous CMP
  - GP-GPUs

SoC and IoT Market Size







15

15

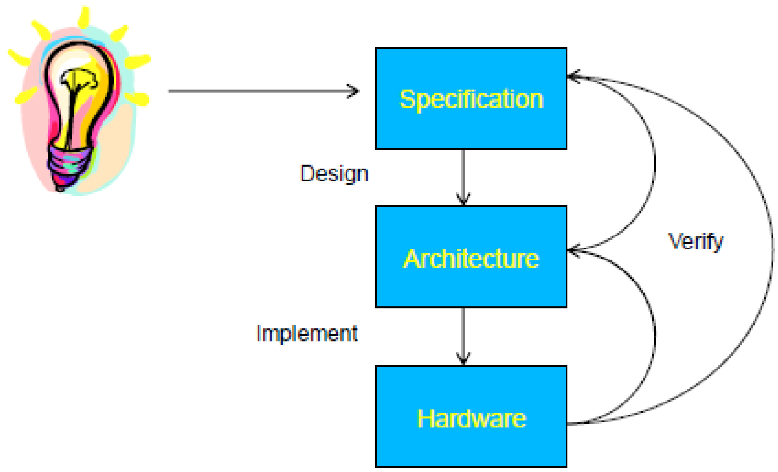
Outline

- What is an Embedded System (ES)
- Examples of embedded systems
- Embedded systems characteristics
- How to design an embedded system
- ARM Cortex-M3

16

16

Design Process

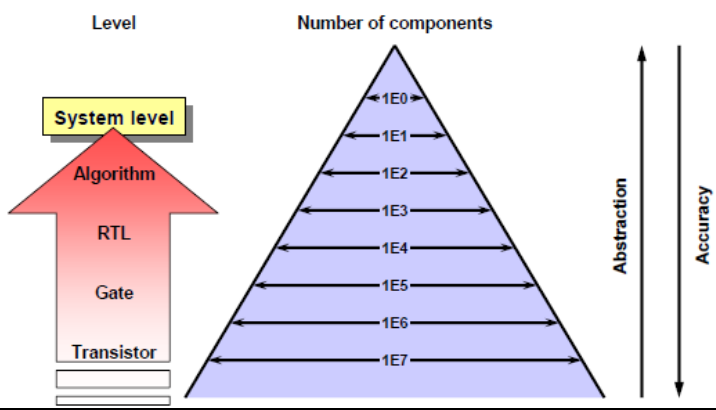


17

17

Challenges: Complexity and Heterogeneity

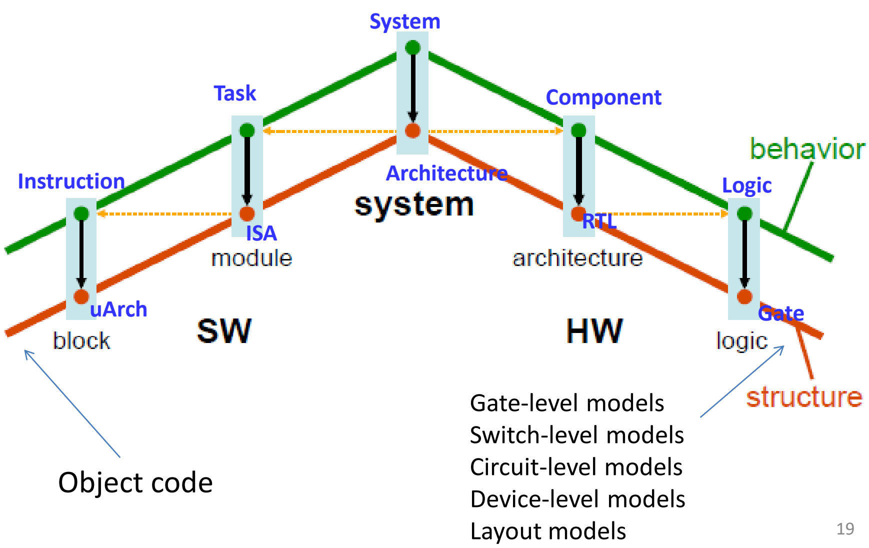
- **Complexity**
  - High degree of parallelism at various levels
  - High degree of design freedom
  - Multiple optimization objectives design constraints
- **Handled by working at higher levels of abstraction, hierarchy**



18

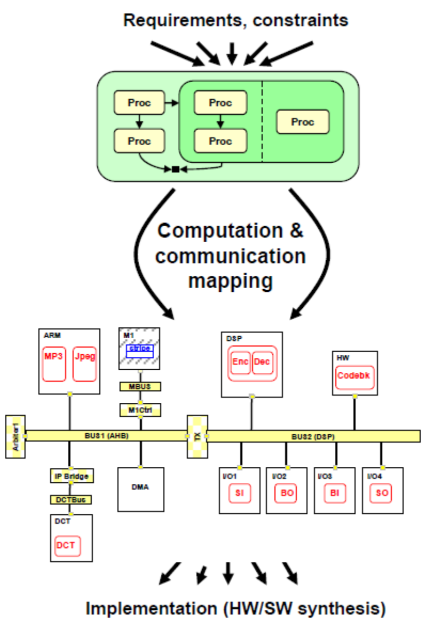
18

Abstraction Layers



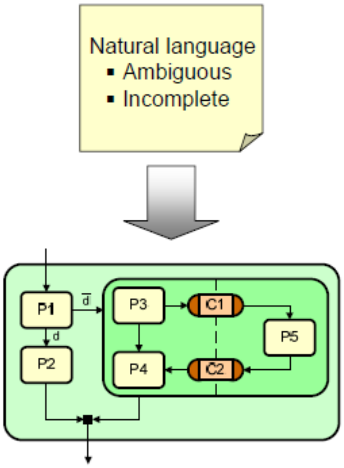
System Level Design

- **From** **specification**
  - Functionality, behavior
    - Application algorithms
    - Constraints
- **To implementation**
  - Architecture
    - Spatial and temporal order
    - Components and connectivity
    - Across hardware and software
- **Design automation at the system level**
  - Modeling & simulation
  - Synthesis & exploration
  - Verification



System Specification

- **Capture requirements (what)**
  - Functional
    - Free of any implementation details
  - Non-functional
    - Constraints
- **Formal representation**
  - Models of computation
    - Objects & composition rules
    - Concurrency & time
    - Computation & communication
  - Executable
    - Semantics
- **Application development**
  - Precise description of desired system behavior
    - Complete and unambiguous

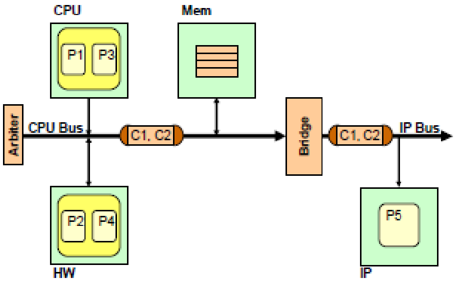


21

21

System Architecture

- **Architecture definition**
  - Processing elements (PEs)
    - Processors, memories, FPGAs, DSPs
  - Communication elements
    - Busses, Networks-on-Chip (NoCs), transducers, bus bridges
- **Virtual platform prototyping**
  - PE simulation (functional, full-system) for computation
  - Event-driven simulation, transaction-level modeling (TLM) for communication



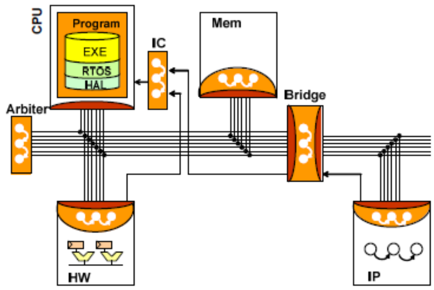
- **Design space exploration and system optimization**
  - Partitioning, mapping (allocation + binding), scheduling
  - Estimation: Synthesis based on abstraction only makes sense if there are powerful estimation methods available:
    - Estimate properties of the next layer(s) of abstraction
    - Design decisions are based on these estimated properties

22

22

System Implementation

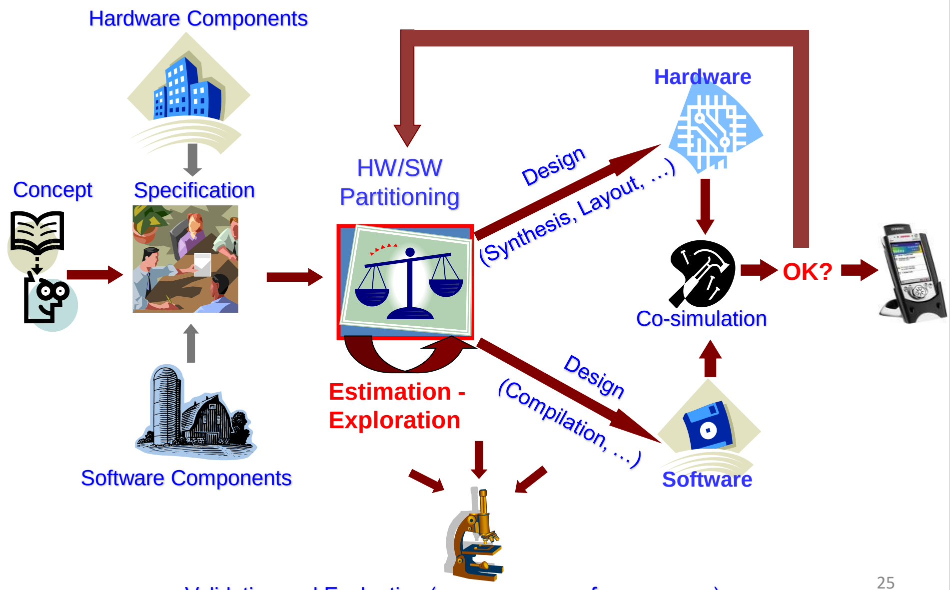
- **Hardware**
  - Microarchitecture models
  - Register-transfer level (RTL)
  - Layouts
- **Software binaries**
  - Application object code
  - Real-time operating system (RTOS)
  - Hardware abstraction layer (HAL)
- **System netlist**
  - Pins and wires
  - Arbiters, muxes, interrupt controllers (ICs), etc.
  - Bus protocol state machines



Hardware vs. Software Modules

- A significant part of the problem is deciding which parts should be in software on programmable processors, and which in specialized hardware
- **Hardware** = functionality implemented via a custom architecture (datapath + FSM)
- **Software** = functionality implemented in software on a programmable processor
- Key differences:
  - Multiplexing
    - software modules multiplexed with others on a processor
    - hardware modules are typically mapped individually on dedicated hardware
  - Concurrency
    - processors usually have one "thread of control"
    - dedicated hardware often has concurrent datapaths

HW/SW Co-design



25

25

HW/SW Co-design

- HW/SW Co-design means the design of a special-purpose system composed of a few application-specific ICs that cooperate with software procedures on general-purpose processors (1994)
- HW/SW Co-design means meeting system-level objectives by exploiting the synergism of hardware and software through their concurrent design (1997)
- HW/SW Co-design tries to increase the predictability of embedded system design by providing **analysis methods** that tell designers if a system meets its performance, power, and size goals and **synthesis methods** that let designers rapidly evaluate many potential design methodologies (2003)
- It moved from an emerging discipline (early ‘90s) to a mainstream technology (today)

26

26

System Level Design Flow (Methodology)

- Past and present:
  - Ad hoc approaches based on earlier experience with similar products, and on manual design
  - HW/SW partitioning decided at the beginning, and then designs proceed separately
- Present and future:
  - From HW/SW co-design to HW/SW co-synthesis
  - Design automation (CAD) tools: very challenging

27

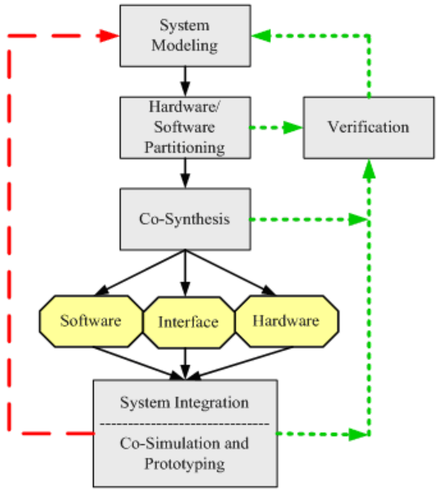
From HW/SW Co-design to HW/SW Co-synthesis!

- Early approaches: HW/SW partitioning would be done first and then HW/SW blocks would be synthesized separately
- Ideally system synthesis would do HW/SW partitioning, mapping, and scheduling in a unified fashion – very difficult
- Design space exploration (estimation and refinement) would also be done in a unified fashion; by working at the same time with both HW and SW modules → **Co-synthesis**
- Key: communication models

28

HW/SW Co-synthesis

- Co-synthesis: Synthesize the software, hardware and interface implementation in a unified fashion. This is done concurrently with as much interaction as possible between the three implementations.



29

29

Outline

- What is an Embedded System (ES)
- Examples of embedded systems
- Embedded systems characteristics
- How to design an embedded system
- ARM Cortex-M3 (focus of this course)

30

30

Why study the ARM architecture (Cortex-M3 in particular)?

- Very popular in industry
- Lots of manufacturers ship ARM based products
  - What differentiates these products? Peripherals!

31

31

Summary

- Embedded systems are everywhere
- Big picture:
  - Embedded systems → SoCs, CPSs
  - System-level design is key
  - Key challenge: optimization of design metrics (which compete with one another)
  - A unified view of hardware and software is necessary
- Focus of this course:
  - We'll focus on rather simple embedded systems; using ARM Cortex-M3 based MCU

32

32

Embedded Systems and You

- As engineers, it is very likely that you will:
  - Design microprocessors and other digital circuits (e.g., ASICs, FPGAs, etc.) to be used in embedded applications
  - Develop algorithms (control, signal processing, etc.) that will be implemented on embedded microprocessors
  - Develop software (e.g., design automation – CAD – tools, RTOS, apps, etc.) for the embedded market
  - Work in application fields that involve an embedded microprocessor
  - Design sensors/actuators (e.g., MEMS devices) that may be used in embedded systems
  - Design and implement complete systems that contain embedded systems
- It is certain that you encounter embedded systems in all aspects of your daily life!

33

33

Skills Needed

- An embedded system application involves a diverse set of skills that extend across traditional disciplinary boundaries, including
  - computer hardware
  - software
  - algorithms
  - interface electronics
  - application domain
- Make engineering tradeoffs that extend across these boundaries

34

34