ELEC 5260/6260/6266
Embedded Computing Systems

Spring 2019
Victor P. Nelson

Text: "Computers as Components, 4th Edition"
Prof. Marilyn Wolf (Georgia Tech)

Course Web Page:
http://www.eng.auburn.edu/~nelsovp/courses/elec5260_6260/

Course Topics (1)

- Embedded system design and implementation
  - The embedded computing space – what is “embedded computing”?
- System design methodologies (including UML)
- Platforms: system-on-chip (SoC), microcontrollers, FPGAs, networks.
  - CPUs for embedded systems (ARM, DSP)
  - ARM Cortex-M4 and “Discovery Kit” development board
- System architectures, applications, methodologies.
  - Hardware, software, system.
- Hierarchical software design for embedded systems

(continued)

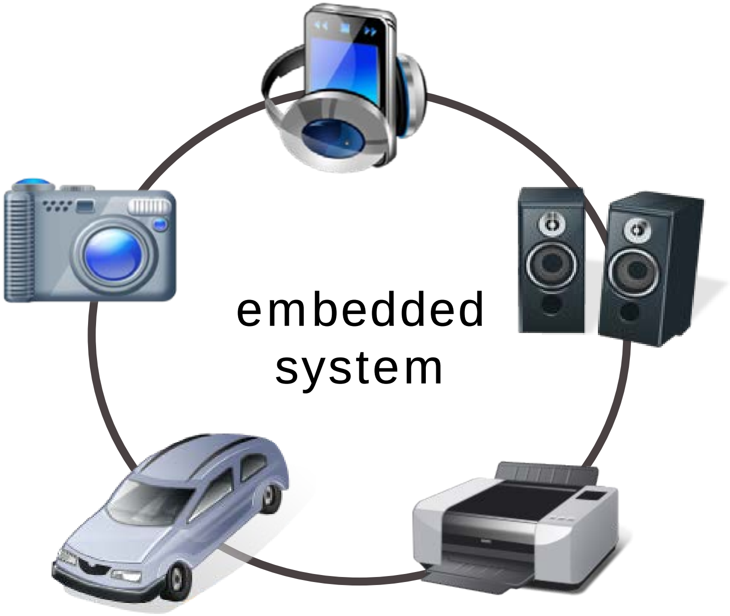
Course Topics (continued)

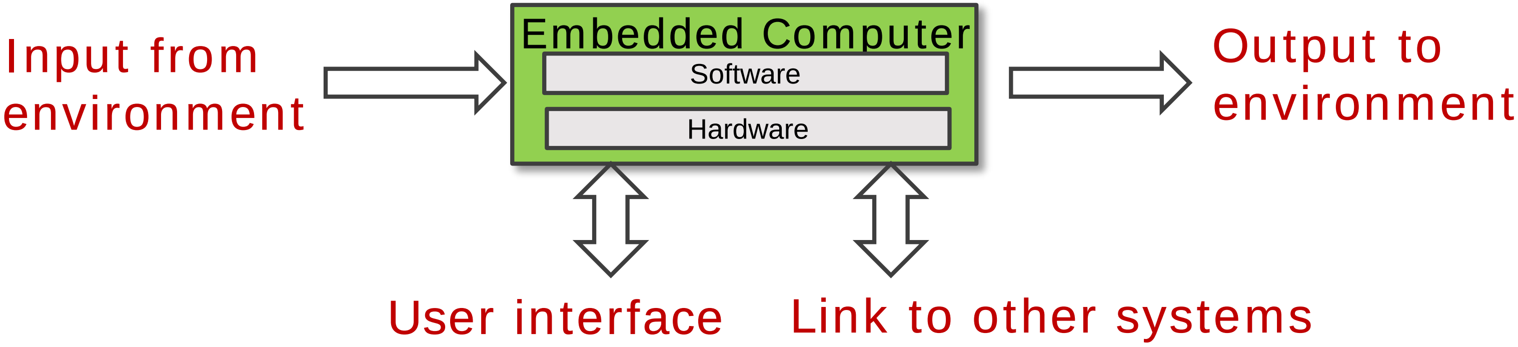
- Input/output devices, interrupts, timing
- Sensors, data acquisition, and control systems
- Real-time operating systems for embedded systems
- Internet of Things, IoT networks
- Automotive and Aerospace systems
- Standards-based design.
- Case studies

This is not simply a “microcontroller course”.

Introduction to embedded systems

- What is an embedded system?
  - *Application-specific* computer system
  - Component of a larger system
  - Interacts with its environment
  - Often has *real-time* computing constraints





Input from environment

Embedded Computer

Software

Hardware

Output to environment

User interface

Link to other systems

Benefits of Embedded Computer Systems

- Greater performance and efficiency
  - Software makes it possible to provide sophisticated control
  - Integrated functions often more efficient than external ones
- Lower costs
  - Less expensive components can be used
  - Manufacturing, operating, and maintenance costs reduced
- More features
  - Many not possible or practical with other approaches
- Better dependability/security
  - Adaptive system which can compensate for failures
  - Better diagnostics to improve repair time
- Potential for distributed system design
  - Multiple processors communicating across a network can lower parts and assembly costs and improve reliability

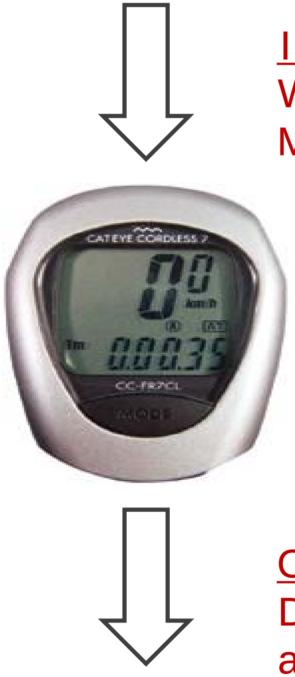
Application examples

- Simple control: microwave oven front panel
- Canon EOS 3 has three microprocessors.
  - 32-bit RISC CPU runs auto-focus and eye control systems.
- Digital TV: programmable CPUs + hardwired logic.
- Smart phone: keyboard, communications, games, app's
- Internet of Things (IoT) - distributed sensors/controllers
- Vehicle control (automotive, aerospace, etc.)
- Industrial process control (nuclear power plant)
- *OTHER EXAMPLES??*

ASSIGNMENT #1: 4-page report on a current multimedia system/device or an IoT system

Example embedded system: bike computer

- Functions
  - Speed and distance measurement
- Constraints
  - Size
  - Cost
  - Power and energy
  - Weight
- Inputs
  - Wheel rotation indicator
  - Mode key
- Output
  - Liquid Crystal Display
- Use Low Performance Microcontroller
  - 8-bit, 10 MIPS



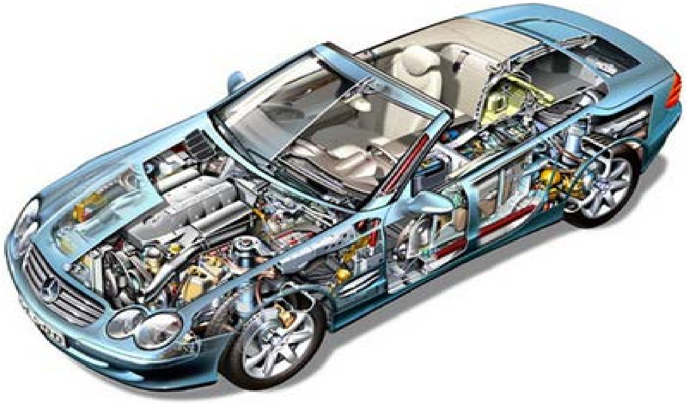
Input:
Wheel rotation
Mode key

Output:
Display speed
and distance

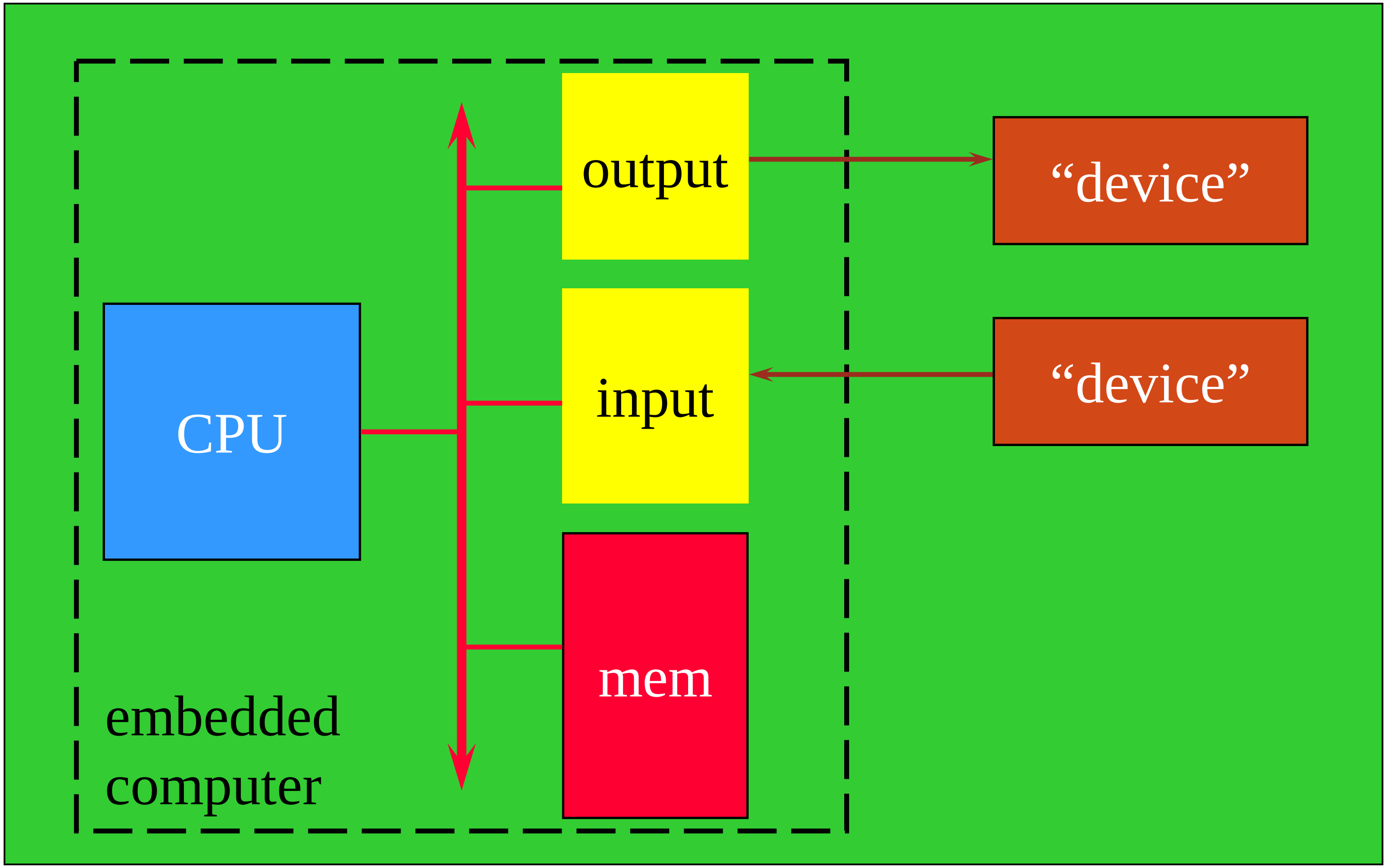
Gasoline automobile engine control unit

- Functions
  - Fuel injection
  - Air intake setting
  - Spark timing
  - Exhaust gas circulation
  - Electronic throttle control
  - Knock control
- Constraints
  - Reliability in harsh environment
  - Cost
  - Weight

- Many inputs and outputs
  - Discrete sensors & actuators
  - Network interface to rest of car
- Use high performance microcontroller
  - e.g. 32-bit, 3 MB flash memory, 150 - 300 MHz



Embedding a computer



Options for Building Embedded Systems

	Implementation	Design Cost	Unit Cost	Upgrades & Bug Fixes	Size	Weight	Power	System Speed
Dedicated Hardware	Discrete Logic	low	mid	hard	large	high	?	very fast
	ASIC	high ($500K/ mask set)	very low	hard	tiny - 1 die	very low	low	extremely fast
	Programmable logic – FPGA, PLD	low	mid	easy	small	low	medium to high	very fast
Software Running on Generic Hardware	Microprocessor + memory + peripherals	low to mid	mid	easy	small to med.	low to moderate	medium	moderate
	Microcontroller (int. memory & peripherals)	low	mid to low	easy	small	low	medium	slow to moderate
	Embedded PC	low	high	easy	medium	moderate to high	medium to high	fast

Microprocessors vs custom circuits?

- Microprocessors can be very efficient:
  - Use same logic to perform many different functions.
  - Create families of products.
  - Create upgradable systems.
- Alternatives:
  - Custom System on Chip (SoC) implemented with ASICs, field-programmable gate arrays (FPGAs), etc.
    - May or may not include microprocessor
  - "Platform" FPGA – implement one or more microprocessor hard/soft cores, with embedded memory and programmable logic

Microprocessor options

- **Microcontroller:** includes I/O devices, on-chip memory.
- **Digital signal processor (DSP):** microprocessor optimized for digital signal processing.
- **Application-Specific Processor (ASP):** instruction set & architecture tailored to application (graphics, network, etc.)
- **Soft core:** microcontroller or CPU model to be synthesized into a system on chip (SoC)
- **Hard core:** microcontroller or CPU implemented as part of a SoC, "platform FPGAs"

Early history

- Late 1940's: MIT Whirlwind computer was designed for real-time operations.
  - Originally designed to control an aircraft simulator.
- HP-35 calculator used several chips to implement a microprocessor in 1972.
- First microprocessor was Intel 4004 in early 1970's.
- 4-bit microcontrollers created in the 1970's
- 8-bit microcontrollers in mid 1970's
- *and so on* …

Early history, continued.

- **Automobiles** have used microprocessor-based engine controllers starting in 1970's.
  - Control fuel/air mixture, engine timing, etc.
  - Multiple modes of operation: warm-up, cruise, hill climbing, etc.
  - Provides lower emissions, better fuel efficiency.
- **High-performance** 32- and 64-bit microcontrollers enable movement of functions from HW to SW
  - Radio.
  - Multimedia.
  - Communications
  - Complex control.
- **Networks** of lower-level microcontrollers distribute tasks

Automotive embedded systems

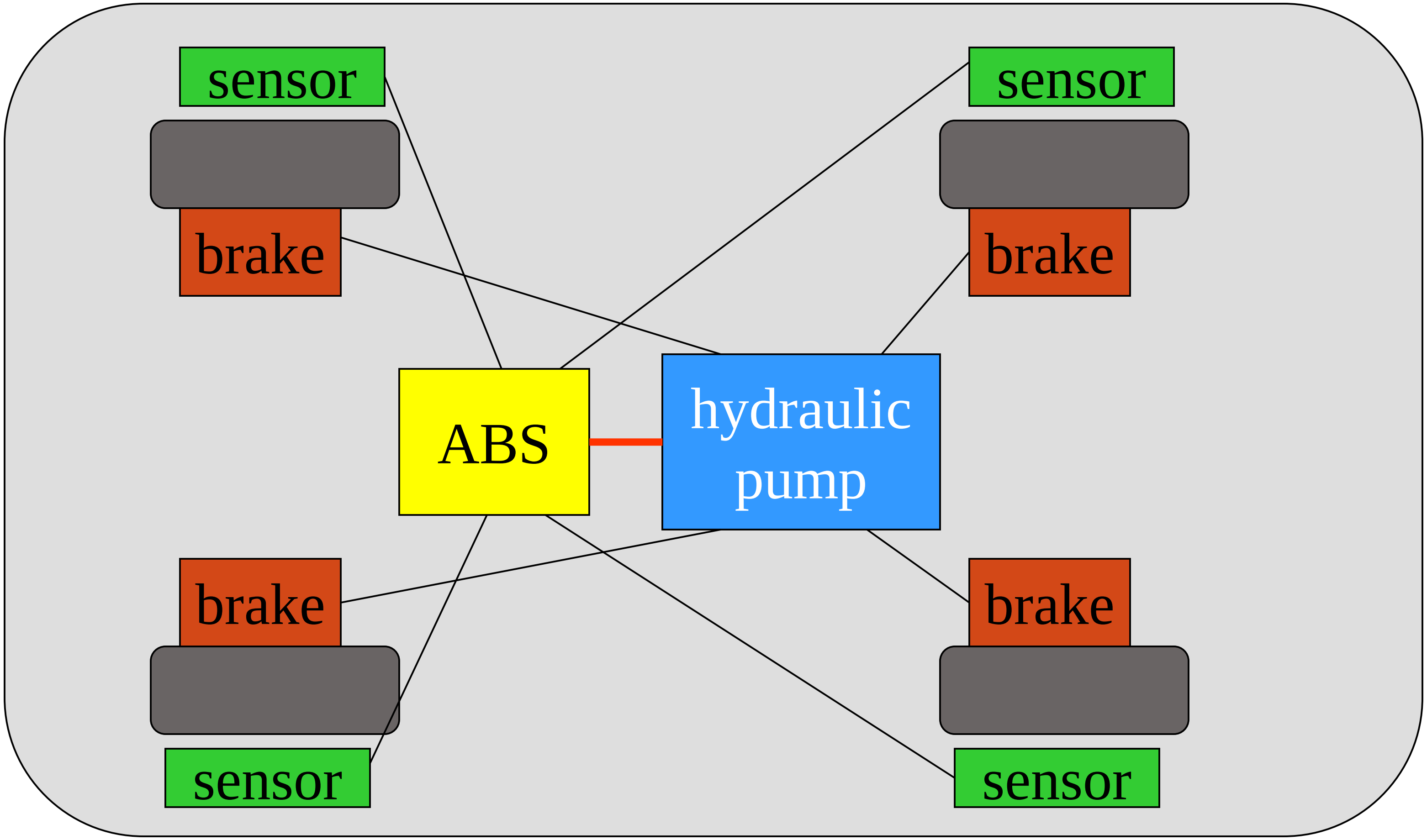
- High-end automobile may have *dozens* of microprocessors:
  - 8-bit microcontroller checks seat belt;
  - Microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.
  - Network of microcontrollers control antilock brakes
  - Entertainment systems
  - Navigation systems
  - Collision avoidance
  - Autonomous operation **(self-driving)**

BMW 850i brake & stability control system

- Anti-lock brake system (ABS)
  - Pump brakes to reduce skidding.
- Automatic stability control + traction (ASC+T)
  - Control engine to improve stability (throttle, ignition timing, differential brake, gears).
- ABS and ASC+T communicate.
  - ABS was introduced first---needed to interface to existing ABS module.

Diagram – next slide

BMW 850i, cont'd.



High-end embedded system characteristics

- Complex algorithms: high performance & functionality.
- High data rates
- Large data structures
- Varied user/device interfaces.
- Multiple tasks, heterogeneous.
- Real-time operation/precise timing.
- Low-power operation.
- Safe, reliable, secure operations.
- Manufacturable, sustainable, cost-effective.

Often have to make **trade-offs** to meet constraints

Challenges in embedded system design

- How much hardware do we need?
    - CPU computing power? Memory?
    - What peripheral functions?
        - Implement in HW or SW?
- How do we meet timing constraints?
    - Faster hardware or cleverer software?
    - Real-time operating system or custom design?
- How do we minimize power consumption?
- How do we optimize cost?
- How do we ensure system security/reliability?
- How do we meet our time-to-market deadline?