What are “embedded systems”?
- Software running on a device that is not ordinarily considered a computer
- Small number of dedicated tasks
- Domains: mobile computing, medical, automotive/aerospace, industrial control, home appliances, military, sensor networks, Cyber-Physical Systems, smart buildings...

What are the challenges of developing embedded systems?
- limited resources: memory, power, CPU speed, bandwidth
- performance/responsiveness requirements
- real-time constraints
- usability
- software upgrades
- debugging
- unanticipated user environments
- reliability, availability, safety
- heterogeneity: hardware and software

How is “embedded systems programming” different from “regular” programming?
- need to consider constraints
- safety and performance are much more critical, even at the cost of reduced “internal quality”

How does embedded systems design differ from “regular” software system design?
- Need to consider hardware as well as software

Codesign: two or more elements of the system are designed together, trading features, costs, advantages and disadvantages of each element against those of each other element. Specifically relevant is codesign of the software with the hardware. (http://www.hpcwire.com/hpcwire/2010-11-02/-compilers_and_more_hardware_software_codesign.html)
- General-purpose processors – more flexibility
- Application-specific instruction set processors (ASIP); microcontroller
- Field programmable gate array (FPGA)
- Application-specific integrated circuit (ASIC) – more performance, power efficiency

Behavioral Specification:
- what does it need to do?
- what else will it need to communicate with?
- performance constraints
- quality: availability/reliability

Physical Specification (Architecture):
- what type of microprocessor? ISA, supported languages...
- how much RAM?
- what peripherals/interfaces?
- physical dimensions
- power restrictions
- cost constraints

Mapping: Assign behavioral functionality to different physical/architectural elements

Synthesis: design/implement the hardware and software (for software, we will use Model-Driven Development, which will be discussed next week)

Testing/Verification
- testing: show that there are defects
- verification: prove that there are no defects

Useful links:
http://www.tik.ee.ethz.ch/education/lectures/hswcd/
http://faculty.cs.tamu.edu/rabi/cpsc689/lectures/lecture01/hwswcodesign.pdf
http://embedded.eecs.berkeley.edu/research/hsc/class/
http://faculty.cs.tamu.edu/rabi/cpsc689/
http://www.seas.upenn.edu/~cis540/