CSE423: Embedded Systems
Summer, 2020

Introduction to Embedded Systems

Today's Lecture

- What is the embedded system?
- Characteristic and application of Embedded Systems
- Use of micro-processor in embedded systems

What is Embedded Systems?

- Definition: An Embedded system is a microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations.

Embedded Systems (Contd.)

- We are surrounded by Embedded Systems.
  - Cell Phones
  - Automatic Washing Machines.
  - Traffic Signals with Timers.
  - Automobile Electronics.

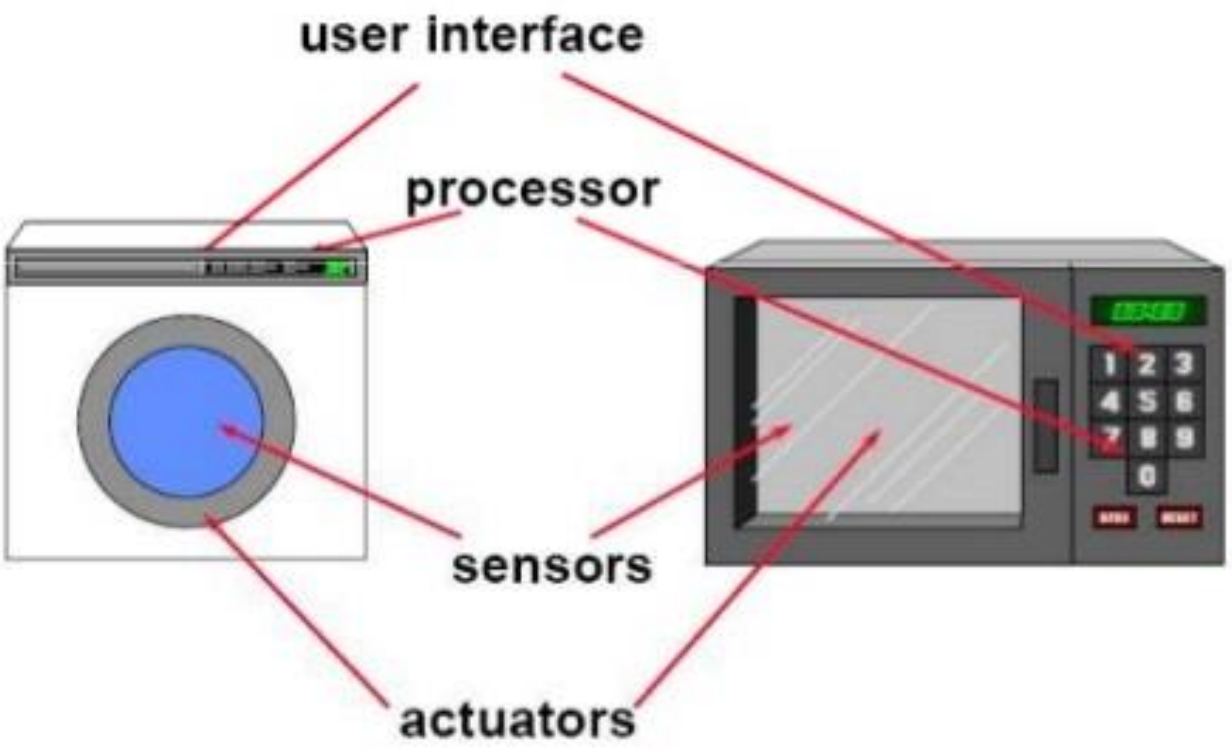
- Find a system that contains no electronic system.
- How can a electronic system improve the functionality/efficiency of that system.
- Custom design an embedded system for the same.

Embedded Systems (contd.)

- Embedded system means the processor is embedded into that application.
- An embedded product uses a microprocessor or microcontroller to do one task only.
- In an embedded system, there is only one application software that is typically burned into ROM.
- Example : printer, keyboard, video game player

Embedded Systems (contd.)

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



Characteristics of Embedded Systems

- The system must meet the following characteristics:
  - Performance
  - Cost/size
  - Real time requirements
  - Power consumption
  - Reliability

Characteristics of Embedded Systems

- Must be ***efficient***:
  - ***Energy*** efficient
  - ***Code-size*** efficient (especially for systems on a chip)
  - ***Run-time*** efficient
  - ***Weight*** efficient
  - ***Cost*** efficient

- ***Dedicated*** towards a certain ***application***: Knowledge about behavior at design time can be used to minimize resources and to maximize robustness.
- ***Dedicated*** user ***interface*** (no mouse, keyboard and screen).

Characteristics of Embedded Systems

- Many ES must meet ***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.
  - For real-time systems, right answers arriving too late (or even too early) are wrong.

> ***„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.

Characteristics of Embedded Systems

- Frequently ***connected to physical environment*** through sensors and actuators,
- ***Hybrid systems*** (analog + digital parts).
- Typically, ES are ***reactive systems***:

> ***„A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment" [Bergé, 1995]***

- Behavior depends on input and current state.
  ☞ automata model often appropriate,

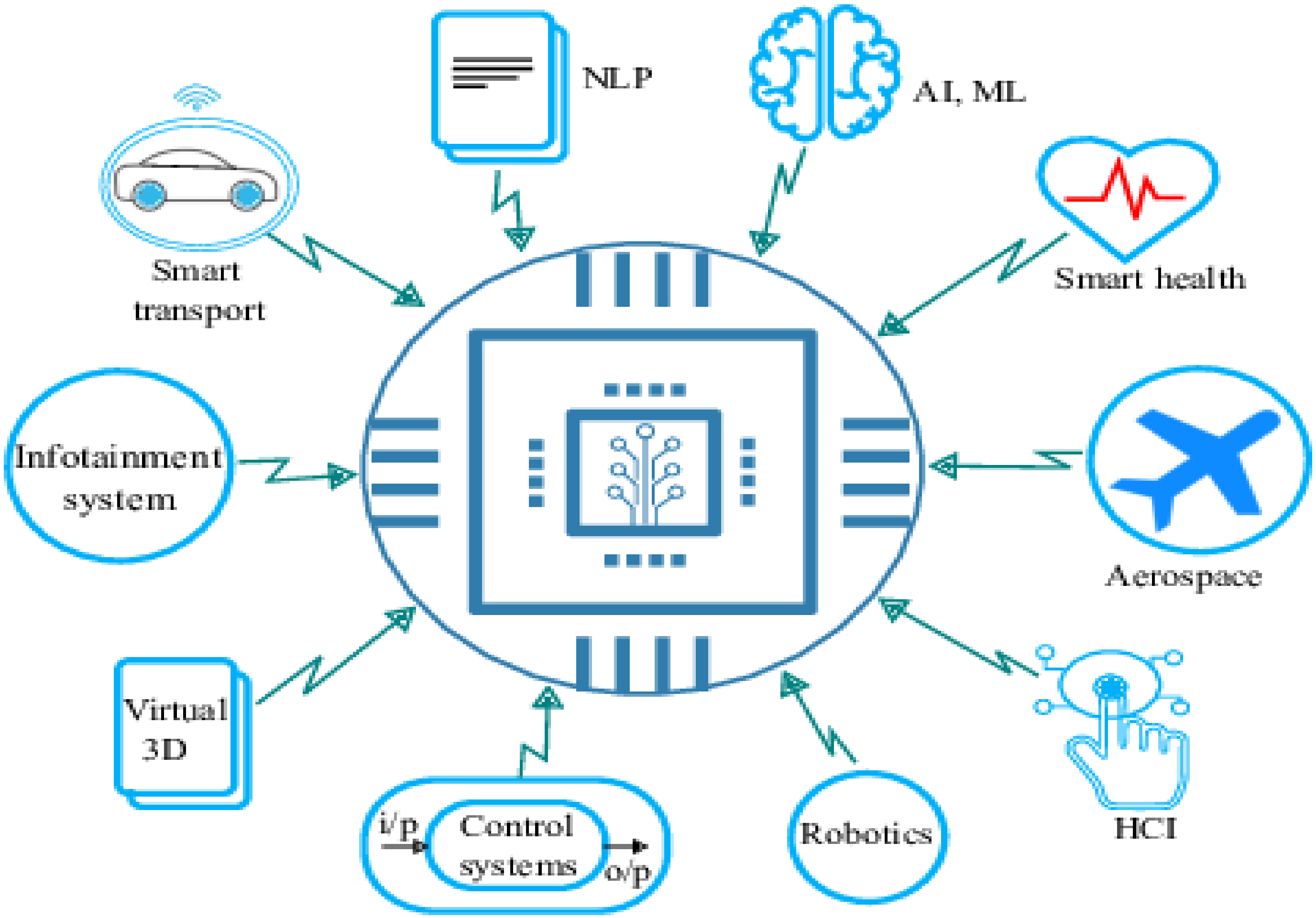
DIUCSE

Characteristics of Embedded Systems

- Embedded Systems
  - Few applications that are known at design-time.
  - Not programmable by end user.
  - Fixed run-time requirements (additional computing power not useful).
  - Criteria:
    - cost
    - power consumption
    - predictability

- General Purpose Computing
  - Broad class of applications.
  - Programmable by end user.
  - Faster is better.
  - Criteria:
    - cost
    - average speed

Current Domain of Embedded Systems



Components of Embedded Systems

- Analog Components
  - Sensors, Actuators, Controllers, …
- Digital Components
  - Processor, Coprocessors
  - Memories
  - Controllers, Buses
  - Application Specific Integrated Circuits (ASIC)
- Converters – A2D, D2A, …
- Software
  - Application Programs
  - Exception Handlers

Microprocessors in Embedded Systems?

- Alternatives: field-programmable gate arrays (FPGAs), custom logic, application specific integrated circuit (ASIC), etc.
- Microprocessors are often very efficient: can use same logic to perform many different functions.
- Microprocessors simplify the design of families of products.

Why Use Microprocessors?

- Two factors that work together to make microprocessor-based designs fast
  - First, microprocessors execute programs very efficiently. While there is overhead that must be paid for interpreting instructions, it can often be hidden by clever utilization of parallelism within the CPU
  - Second, microprocessor manufacturers spend a great deal of money to make their CPUs run very fast.

Why Use Microprocessors?

- Performance
  - Microprocessors use much more logic to implement a function than does custom logic.
  - But microprocessors are often at least as fast:
    - heavily pipelined;
    - large design teams;
    - aggressive VLSI technology.
- Power consumption
  - Custom logic is a clear winner for low power devices.
  - Modern microprocessors offer features to help control power consumption.
  - Software design techniques can help reduce power consumption.
- Heterogeneous systems: some custom logic for well-defined functions, CPUs+ software for everything else.

Microprocessor Varieties

- Microcontroller: includes I/O devices, on-board memory.
- Digital signal processor (DSP): microprocessor optimized for digital signal processing.
- Typical embedded word sizes: 8-bit, 16-bit, 32-bit.

Microprocessor Varieties

- 4-bit, 8-bit, 16-bit, 32-bit :
  - 8-bit processor : more than 3 billion new chips per year
  - 32-bit microprocessors : PowerPC, 68k, MIPS, and ARM chips.
  - ARM-based chips alone do about triple the volume that Intel and AMD peddle to PC makers.
- Most (98% or so) 32-bit processors are used in embedded systems, not PCs.
- RISC-type processor owns most of the overall embedded market [MPF: 2002].

Platforms

- Embedded computing platform: hardware architecture + associated software.
- Many platforms are multiprocessors.
- Examples:
  - Single-chip multiprocessors for cell phone baseband.
  - Automotive network + processors.

The physics of software

- Computing is a physical act.
  - Software doesn’t do anything without hardware.
- Executing software consumes energy, requires time.
- To understand the dynamics of software (time, energy), we need to characterize the platform on which the software runs.

Challenges in Embedded Computing System Design

- How much hardware do we need?
- How do we meet deadlines?
- How do we minimize power consumption?
- How do we design for upgradability?
- Does it really work?

What does “Performance” mean?

- In general-purpose computing, performance often means average-case, may not be well-defined.
- In real-time systems, performance means meeting deadlines.
  - Missing the deadline by even a little is bad.
  - Finishing ahead of the deadline may not help.

Characterizing performance

- We need to analyze the system at several levels of abstraction to understand performance:
  - CPU: microprocessor architecture.
  - Platform: bus, I/O devices.
  - Program: implementation, structure.
  - Task: multitasking, interaction between tasks.
  - Multiprocessor: interaction between processors.

Characteristics of Embedded Systems

- Very high performance, sophisticated functionality
  - Vision + compression + speech + networking all on the same platform.
- Multiple task, heterogeneous.
- Real-time.
- Often low power.
- Low manufacturing cost..
- Highly reliable.
  - I reboot my piano every 4 months, my PC every day.
- Designed to tight deadlines by small teams.

Functional Complexity

- Often have to run sophisticated algorithms or multiple algorithms.
  - Cell phone, laser printer.
- Often provide sophisticated user interfaces.

Design methodologies

- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn’t skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - help automate methodology steps;
  - keep track of the methodology itself.

Levels of abstraction

requirements

specification

architecture

component design

system integration

Our requirements form

name

purpose

inputs

outputs

functions

performance

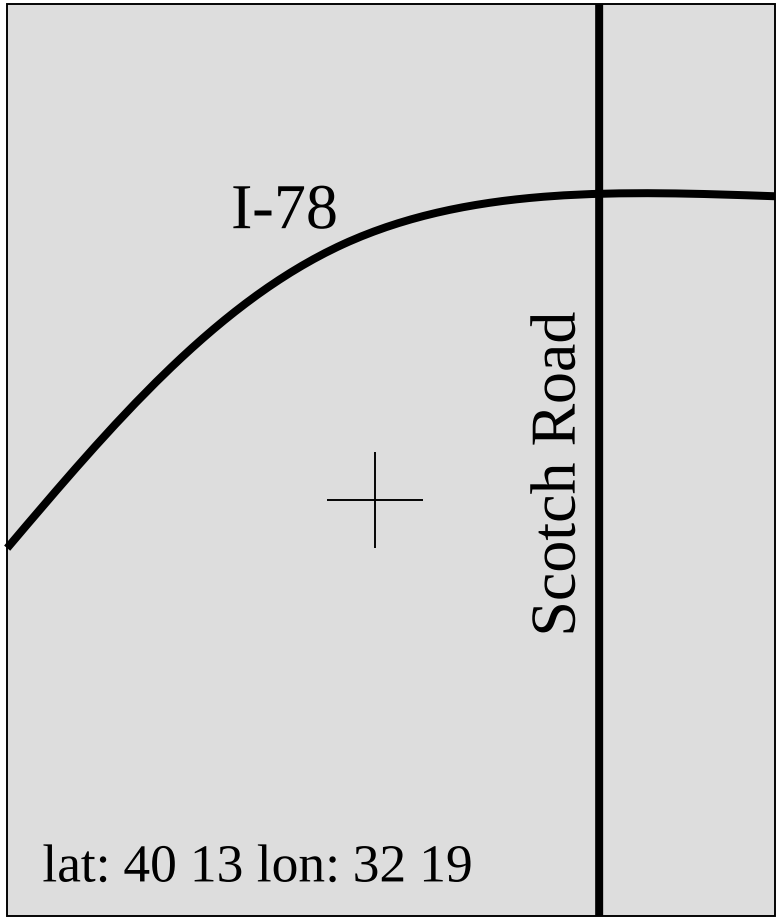
manufacturing cost

power

physical size/weight

Example: GPS moving map requirements

- Moving map obtains position from GPS, paints map from local database.



GPS moving map needs

- Functionality: For automotive use. Show major roads and landmarks.
- User interface: At least 400 x 600 pixel screen. Three buttons max. Pop-up menu.
- Performance: Map should scroll smoothly. No more than 1 sec power-up. Lock onto GPS within 15 seconds.
- Cost: $120 street price = approx. $30 cost of goods sold.

GPS moving map needs, cont'd.

- Physical size/weight: Should fit in hand.
- Power consumption: Should run for 8 hours on four AA batteries.

GPS moving map requirements form

name	GPS moving map
purpose	consumer-grade moving map for driving
inputs	power button, two control buttons
outputs	back-lit LCD 400 X 600
functions	5-receiver GPS; three resolutions; displays current lat/lon
performance	updates screen within 0.25 sec of movement
manufacturing cost	$100 cost-of-goods-sold
power	100 mW
physical size/weight	no more than 2: X 6:, 12 oz.

Specification

- A more precise description of the system:
  - should not imply a particular architecture;
  - provides input to the architecture design process.
- May include functional and non-functional elements.
- May be executable or may be in mathematical form for proofs.

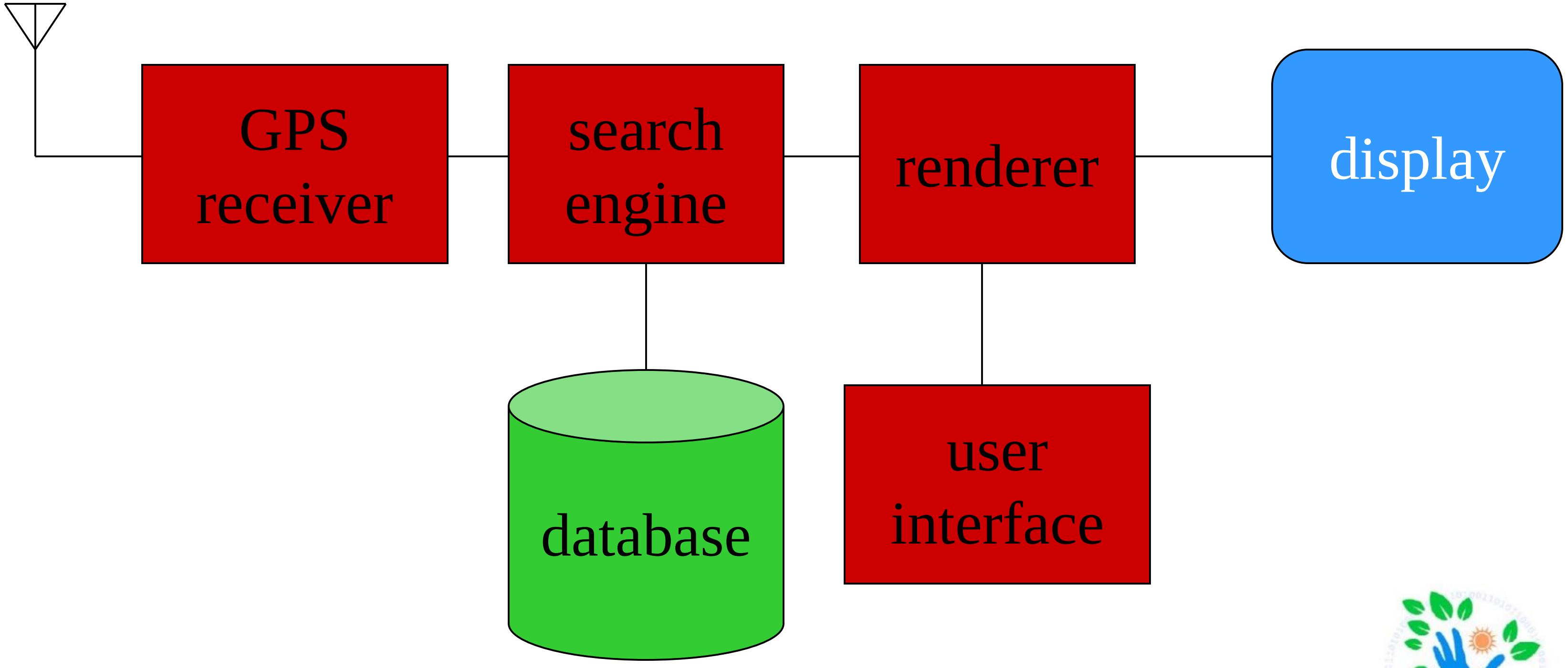
GPS specification

- Should include:
  - What is received from GPS;
  - map data;
  - user interface;
  - operations required to satisfy user requests;
  - background operations needed to keep the system running.

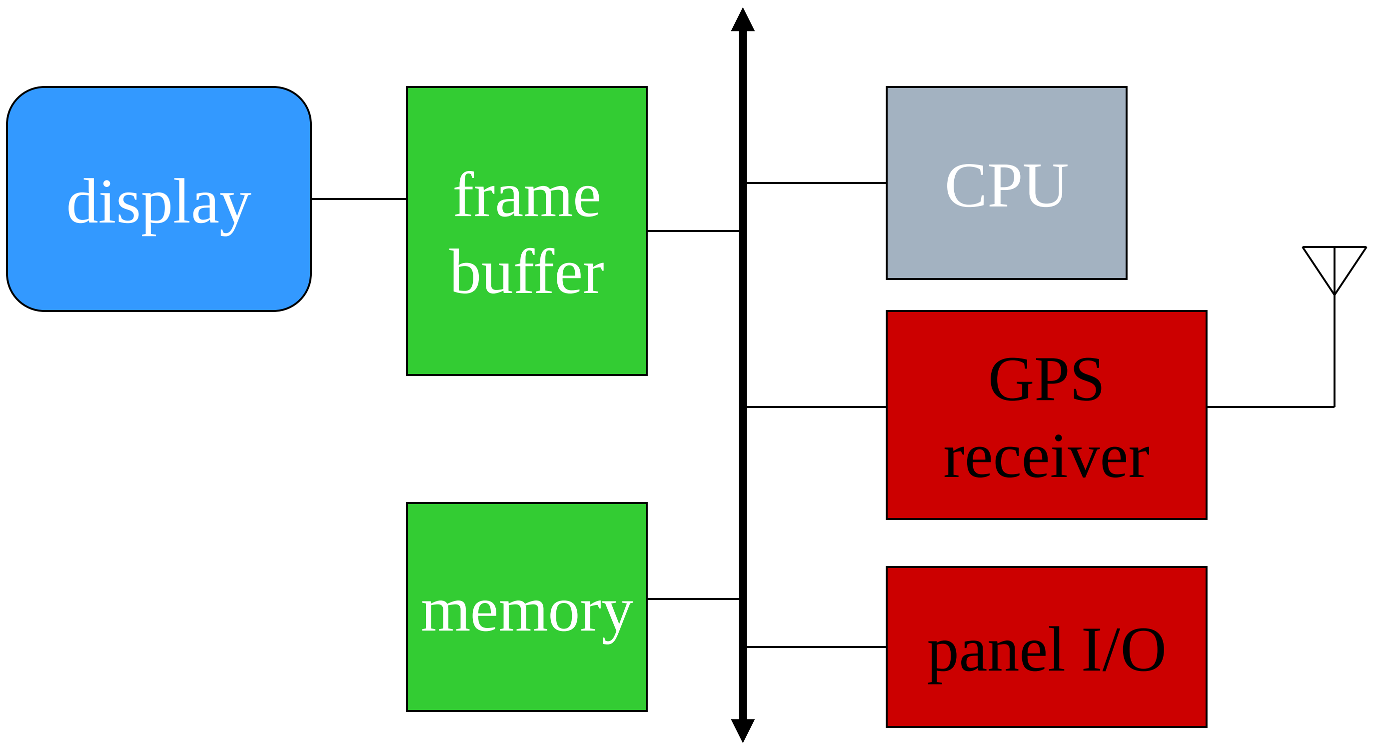
Architecture design

- What major components go satisfying the specification?
- Hardware components:
  - CPUs, peripherals, etc.
- Software components:
  - major programs and their operations.
- Must take into account functional and non-functional specifications.

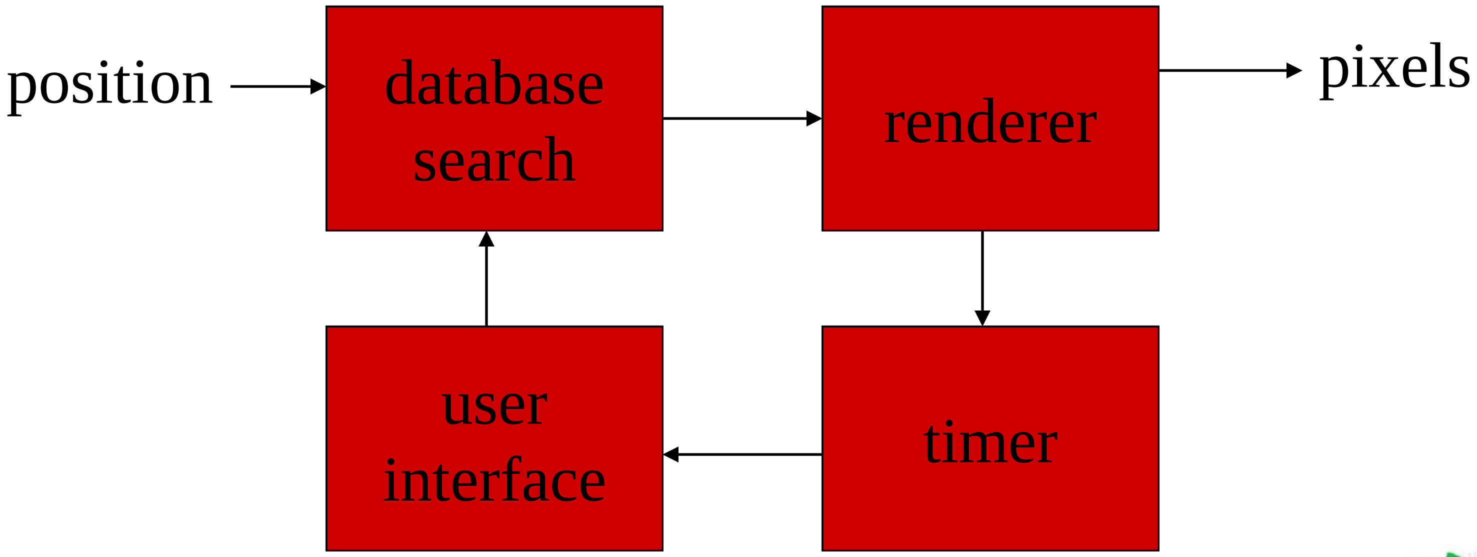
GPS moving map block diagram



GPS moving map hardware architecture



GPS moving map software architecture



Designing hardware and software components

- Must spend time architecting the system before you start coding.
- Some components are ready-made, some can be modified from existing designs, others must be designed from scratch.

System integration

- Put together the components.
  - Many bugs appear only at this stage.
- Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible.

Summary

- Embedded computers are all around us.
  - Many systems have complex embedded hardware and software.
- Embedded systems pose many design challenges: design time, deadlines, power, etc.
- Design methodologies help us manage the design process.