Introduction

- What are embedded computing systems?
- Challenges in embedded computing system design.
- Design methodologies.

Definition

- **A system is an arrangement in which all its units assemble and work together according to the plan or program.**

System Examples

WATCH

It is a time display SYSTEM

Parts: Hardware, Needles, Battery, Dial, Chassis and Strap

Rules

1. All needles move clockwise only
2. A thin needle rotates every second
3. A long needle rotates every minute
4. A short needle rotates every hour
5. All needles return to the original position after 12 hours

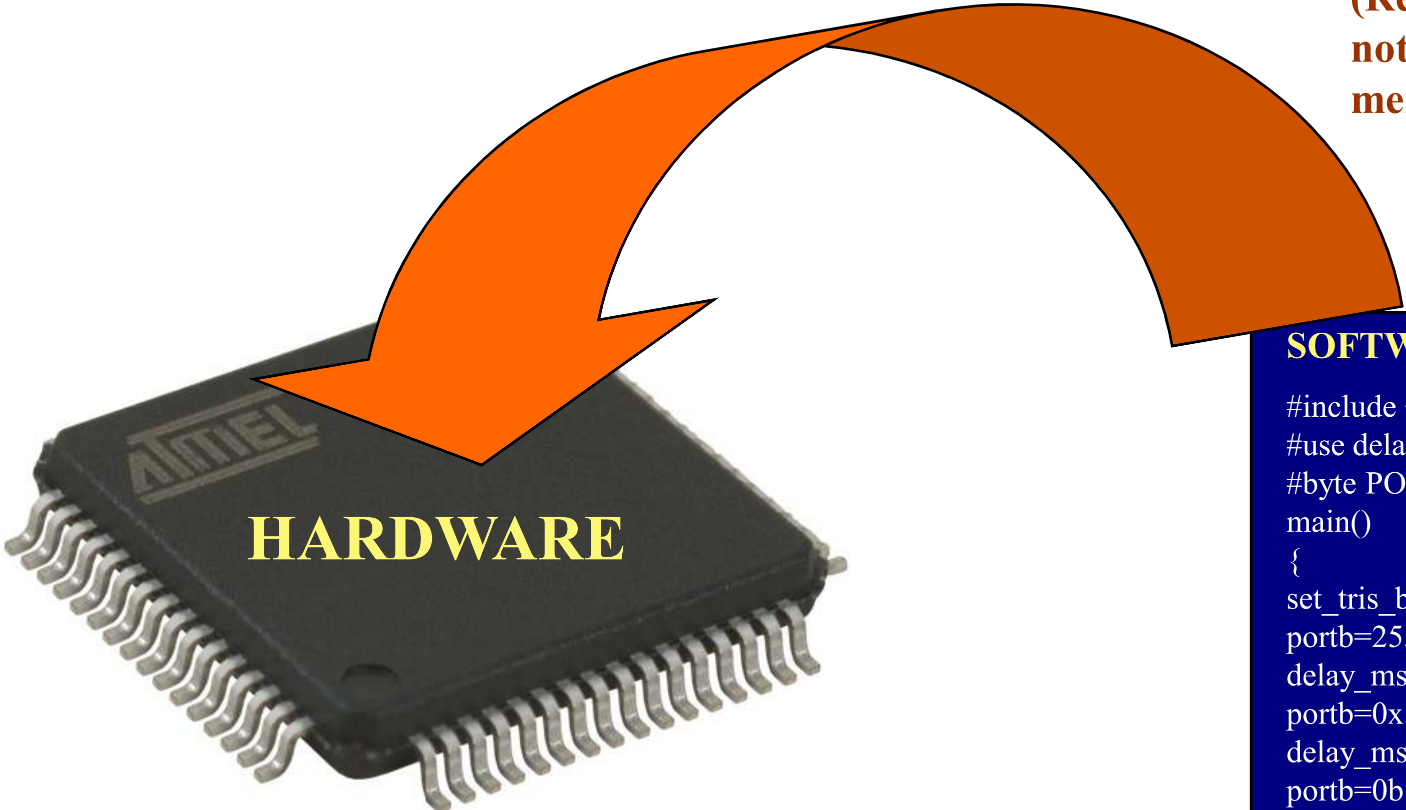
Definition

- ⌘Embedded computing system: Any device that includes a programmable computer but is not itself a general-purpose computer.
- ⌘Embedded systems: Any stand alone computing devices or centers usually dedicated to perform limited computing functions reliably, securely and with minimum upkeep costs.

Embedded System

Definition: An Embedded System is one that has computer hardware with software embedded in it as one of its important components.

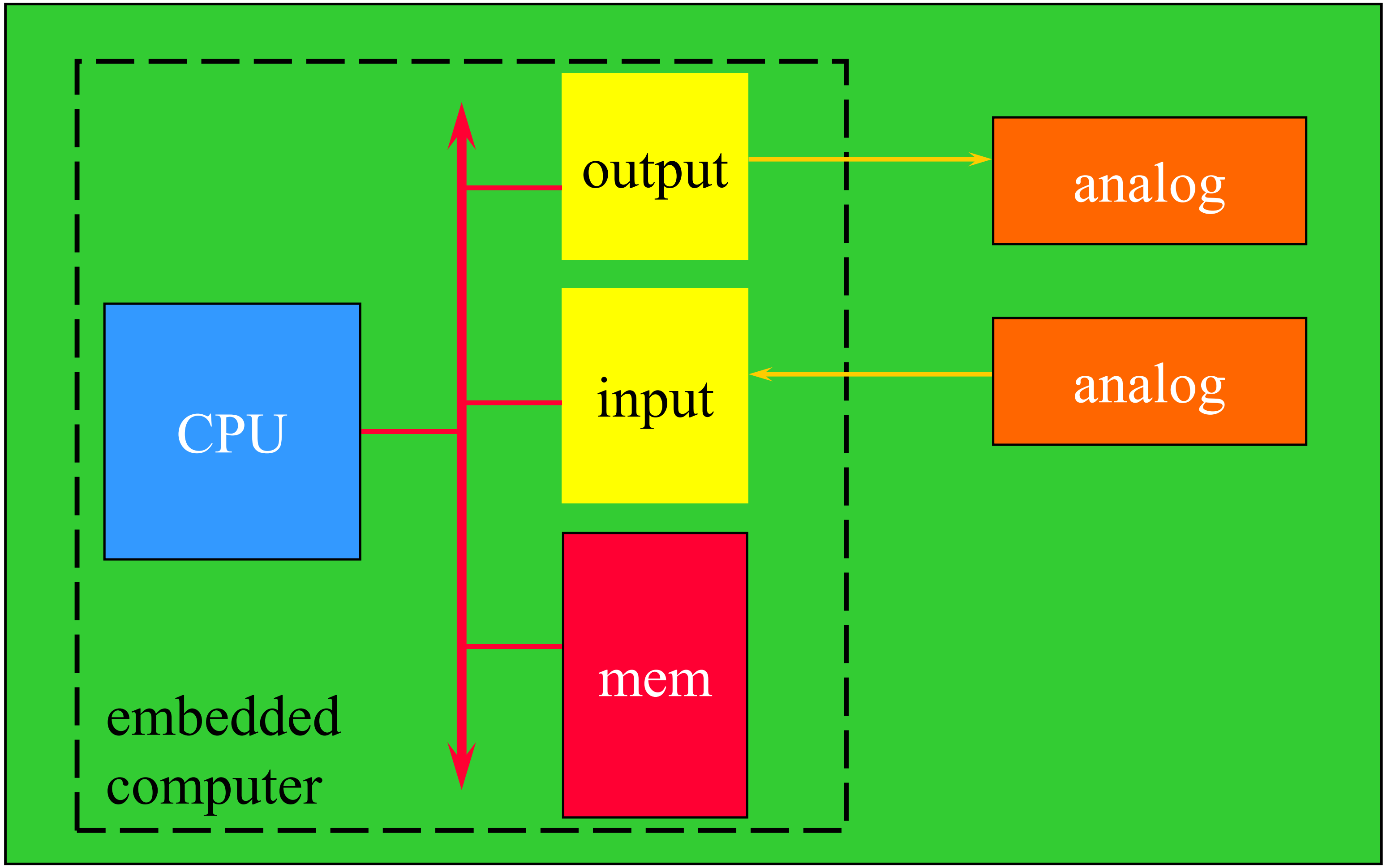
Its software embeds in ROM (Read Only Memory). It does not need secondary memories as in a computer



SOFTWARE PROGRAM

```
#include <16f876a.h>
#use delay (clock=20000000)
#byte PORTB=6
main()
{
set_tris_b(0);
portb=255;       //decimal
delay_ms(1000);
portb=0x55;      //hexadecimal
delay_ms(1000);
portb=0b10101010;  //binary
delay_ms(500);
}
```

Embedding a computer



Components Of Embedded System

- **Hardware**

  Processor, Timers, Interrupt controller, I/O Devices, Memories, Ports, etc.

- **Application Software**

  Which may perform concurrently the series of tasks or multiple tasks.

- **Real Time Operating System (RTOS)**

  RTOS defines the way the system work. Which supervise the application software. It sets the rules during the execution of the application program. A small scale embedded system may not need an RTOS.

Examples

- Cell phone.
- Printer.
- Automobile: engine, brakes, dash, etc.
- Airplane: engine, flight controls etc
- Household appliances.

Early history

- Late 1940's: MIT Whirlwind computer was designed for real-time operations.
  - Originally designed to control an aircraft simulator.
- First microprocessor was Intel 4004 in early 1970's.
  - Designed for an embedded application, namely, a calculator.

Early history, cont'd.

- Automobiles used microprocessor-based engine controllers starting in 1970's.
  - Control fuel
  - Multiple modes of operation: warm-up, hill climbing, etc.
  - Provides lower emissions, better fuel efficiency.

Microprocessor

- Heart of Embedded system.
- **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.

Application examples

- Simple control: front panel of microwave oven, etc.
- Canon EOS 3 has three microprocessors.
  - 32-bit RISC CPU runs autofocus and eye control systems.
- Digital TV: programmable CPUs + hardwired logic for video/audio decode, menus, etc.

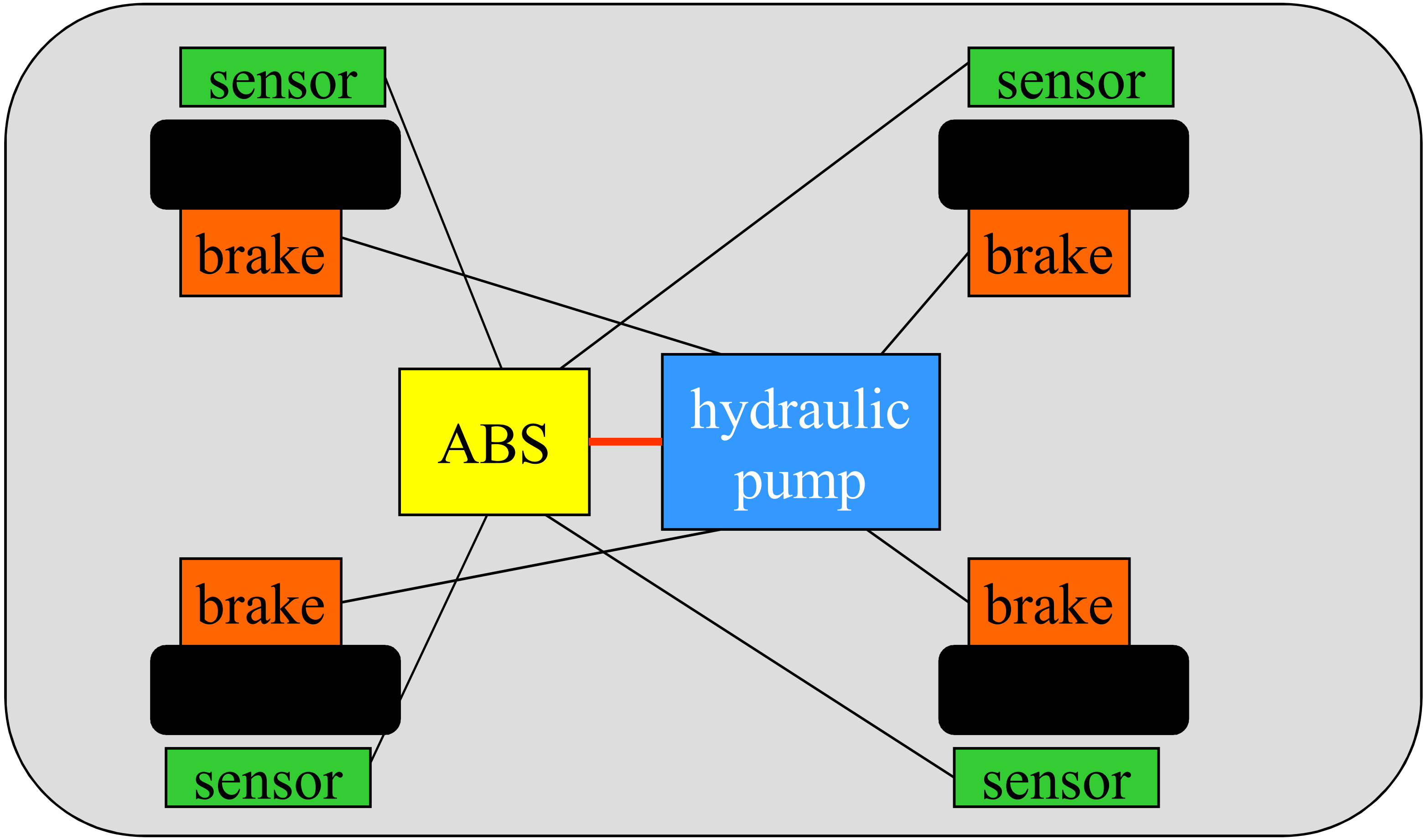
Automotive embedded systems

- Today's high-end automobile may have 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.

BMW 850i brake and stability control system

- Anti-lock brake system (ABS): pumps brakes to reduce skidding.
- Automatic stability control (ASC+T): controls engine to improve stability.
- ABS and ASC+T communicate.

BMW 850i, cont'd.



Characteristics of embedded systems

- Sophisticated functionality.
- Real-time operation.
- Low manufacturing cost.
- Low power.
- 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.

Real-time operation

- Must finish operations by deadlines.
  - Hard real time: missing deadline causes failure.
  - Soft real time: missing deadline results in degraded performance.
- Many systems are multi-rate: must handle operations at widely varying rates.

Non-functional requirements

- Many embedded systems are mass-market items that must have low manufacturing costs.
  - Limited memory, microprocessor power, etc.
- Power consumption is critical in battery-powered devices.
  - Excessive power consumption increases system cost.

Design teams

- Often designed by a small team of designers.
- Often must meet tight deadlines.

Why use microprocessors?

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

The performance paradox

- 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.

Challenges in embedded system design

- How much hardware do we need?
  - How big is the CPU? Memory?
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
- How do we minimize power?
  - Turn off unnecessary logic? Reduce memory accesses?

Challenges, etc.

- Does it really work?
  - Is the specification correct?
  - Does the implementation meet the spec?
  - How do we test for real-time characteristics?
  - How do we test on real data?
- How do we work on the system?
  - Observability, controllability?
  - What is our development platform?

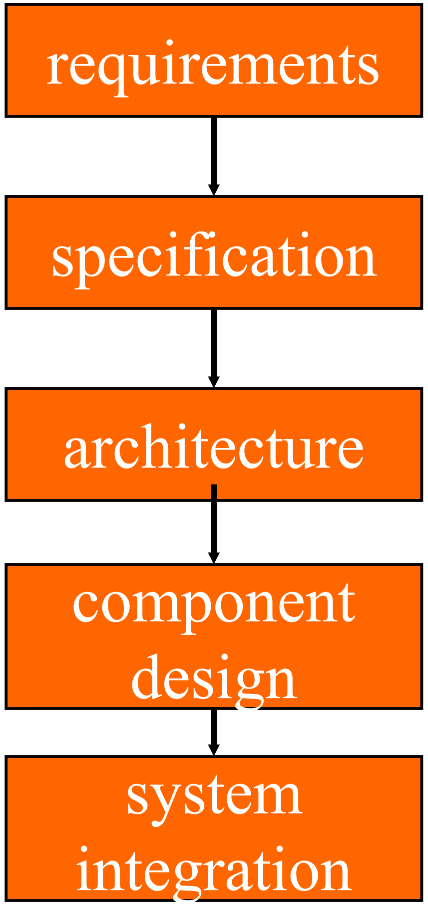
Design methodologies

- 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.

Design goals

- Performance.
  - Overall speed, deadlines.
- Functionality and user interface.
- Manufacturing cost.
- Power consumption.
- Other requirements (physical size, etc.)

Levels of abstraction



Requirements

- Plain language description of what the user wants and expects to get.
- May be developed in several ways:
  - talking directly to customers;
  - talking to marketing representatives;
  - providing prototypes to users for comment.

Functional vs. non-functional requirements

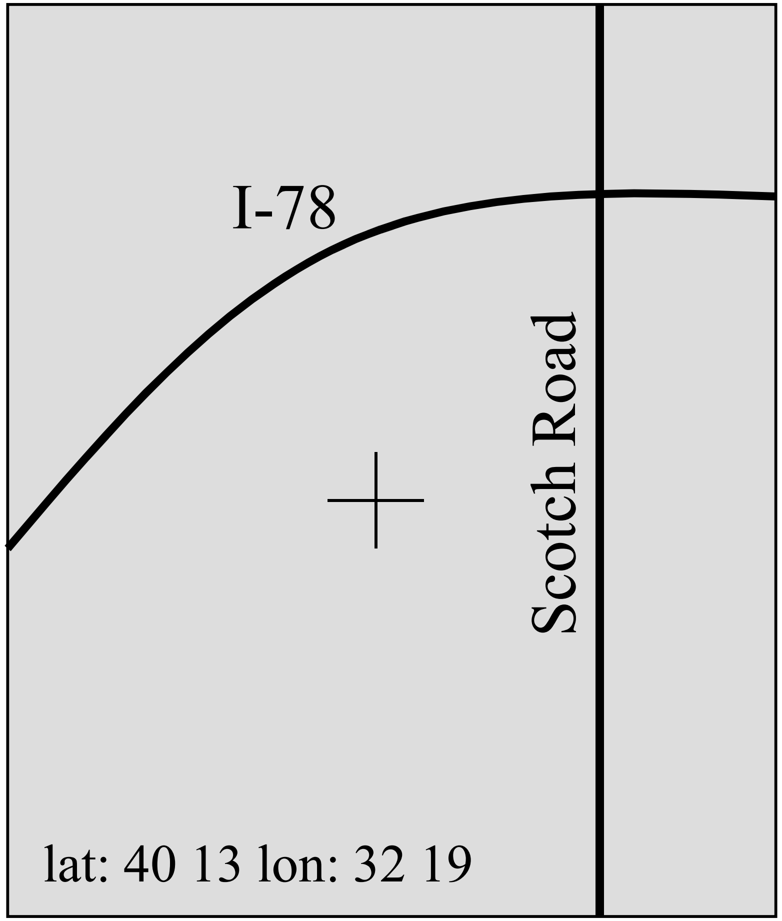
- Functional requirements:
  - output as a function of input.
- Non-functional requirements:
  - time required to compute output;
  - size, weight, etc.;
  - power consumption;
  - reliability;
  - etc.

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.
  - Display should not be more than 1 sec power-up.
  - Verify its position and display the current map within 15 s.
- Cost: Not more than $100.

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 use
Inputs	Power button, two control buttons
Outputs	Back-lit LCD display 400 × 600
Functions	Uses 5-receiver GPS system; three user-selectable resolutions; always displays current latitude and longitude
Performance	Updates screen within 0.25 seconds upon movement
Manufacturing cost	$30
Power	100 mW
Physical size and weight	No more than 2" × 6, " 12 ounces

Specification

- A more precise description of the system:
  - Serves as the contract between the customer and the architects.
  - 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.

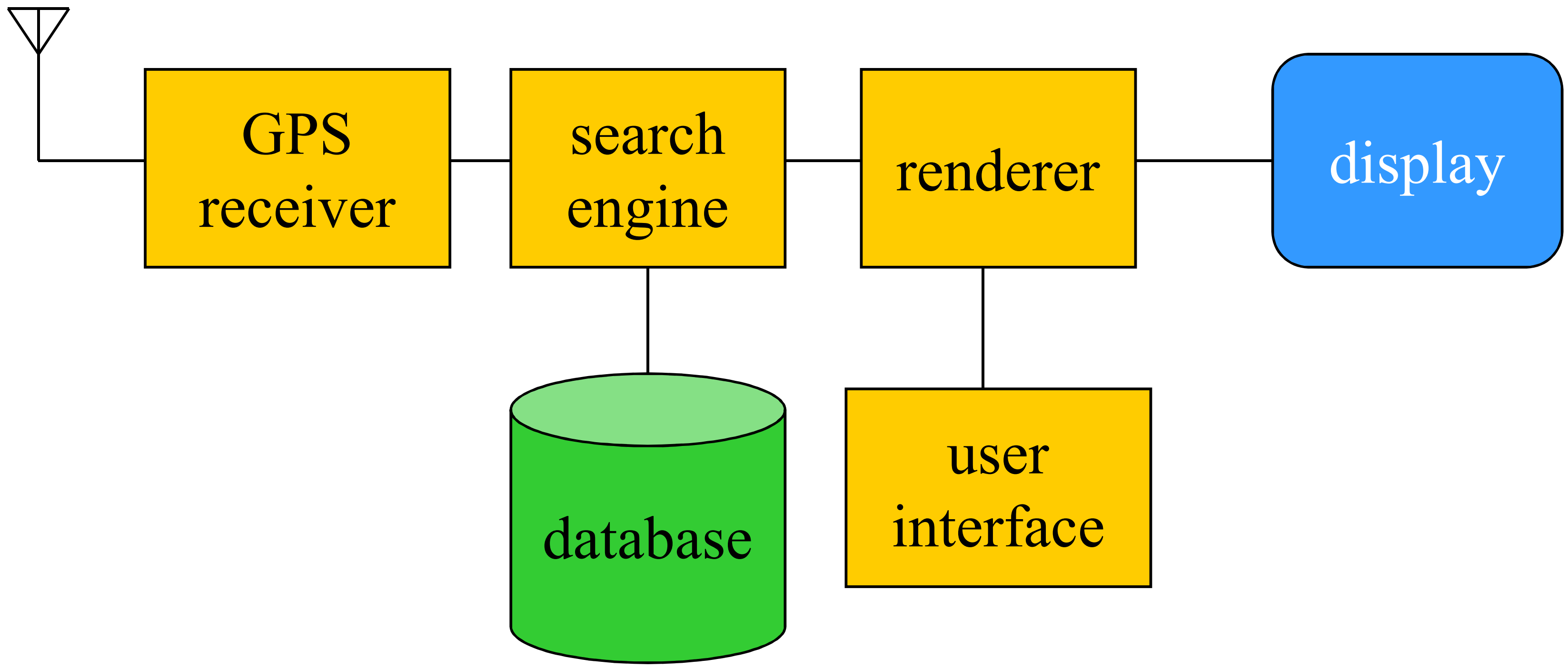
GPS specification

- 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.

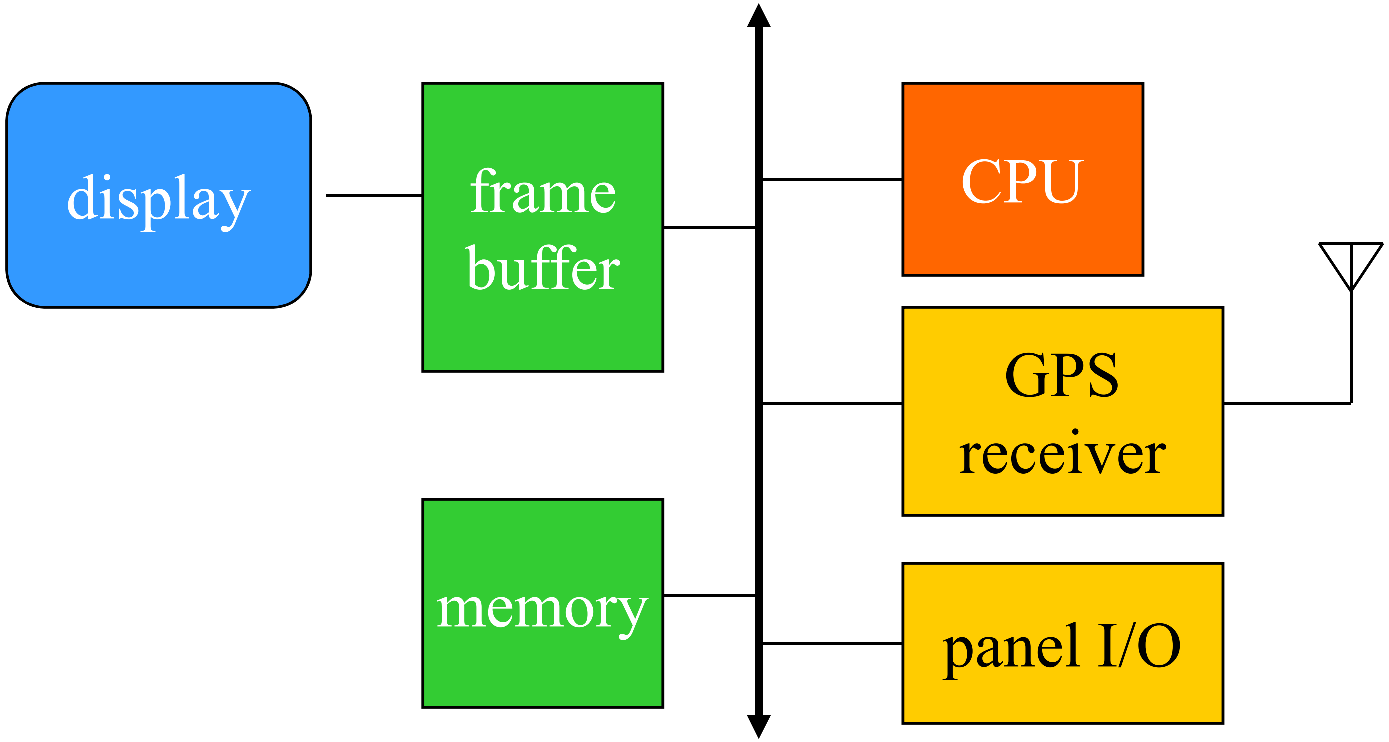
Architecture design

- 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.

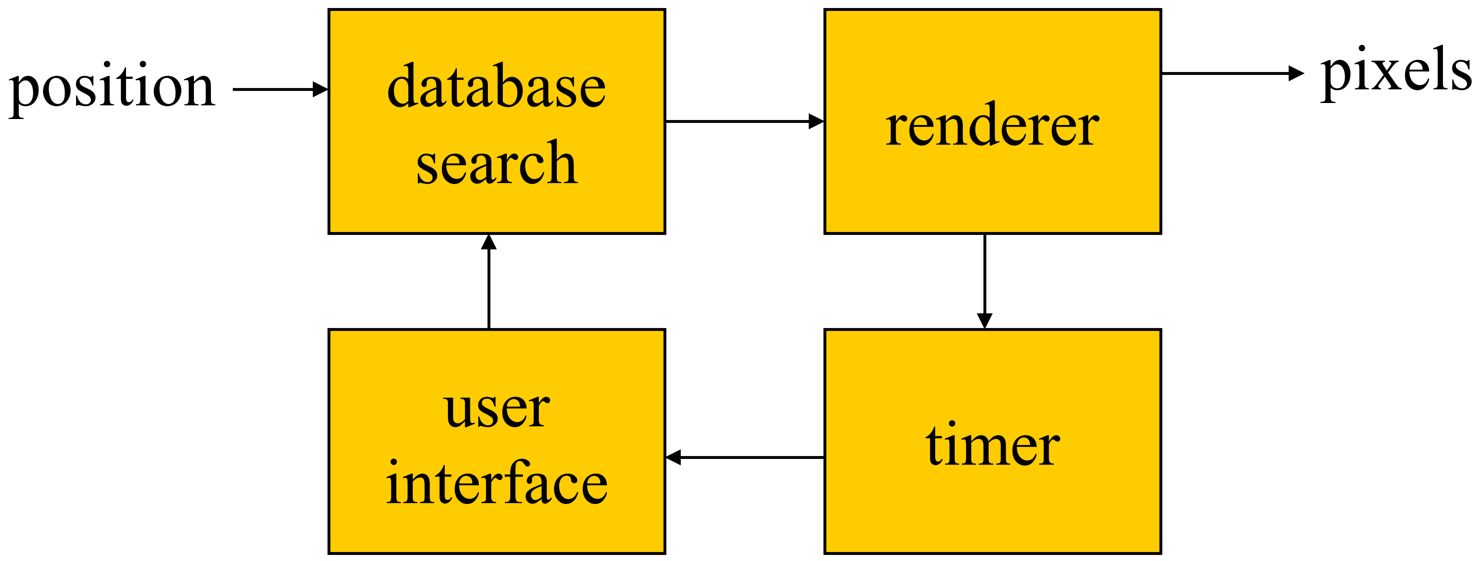
GPS moving map block diagram



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.

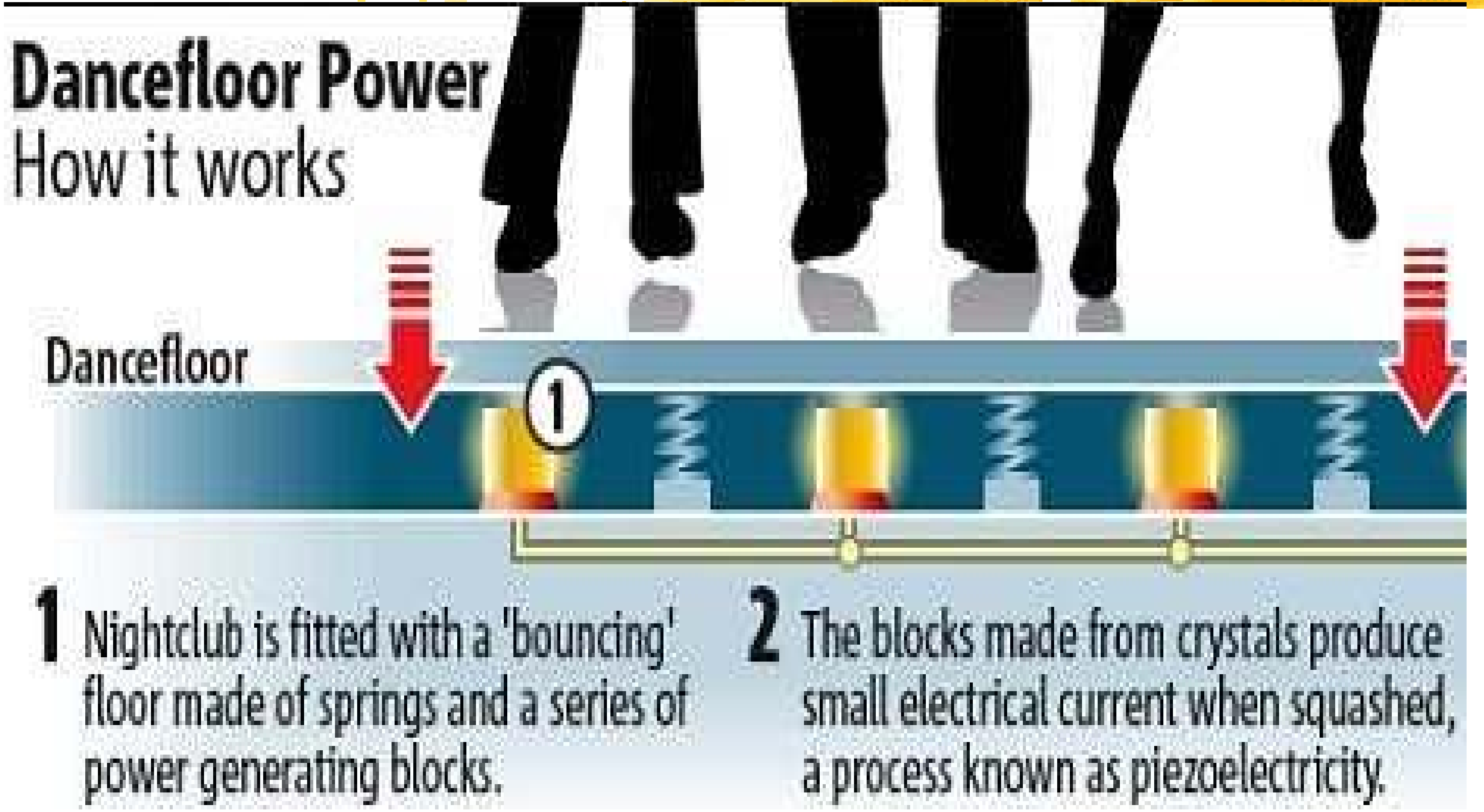
System integration

- 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.

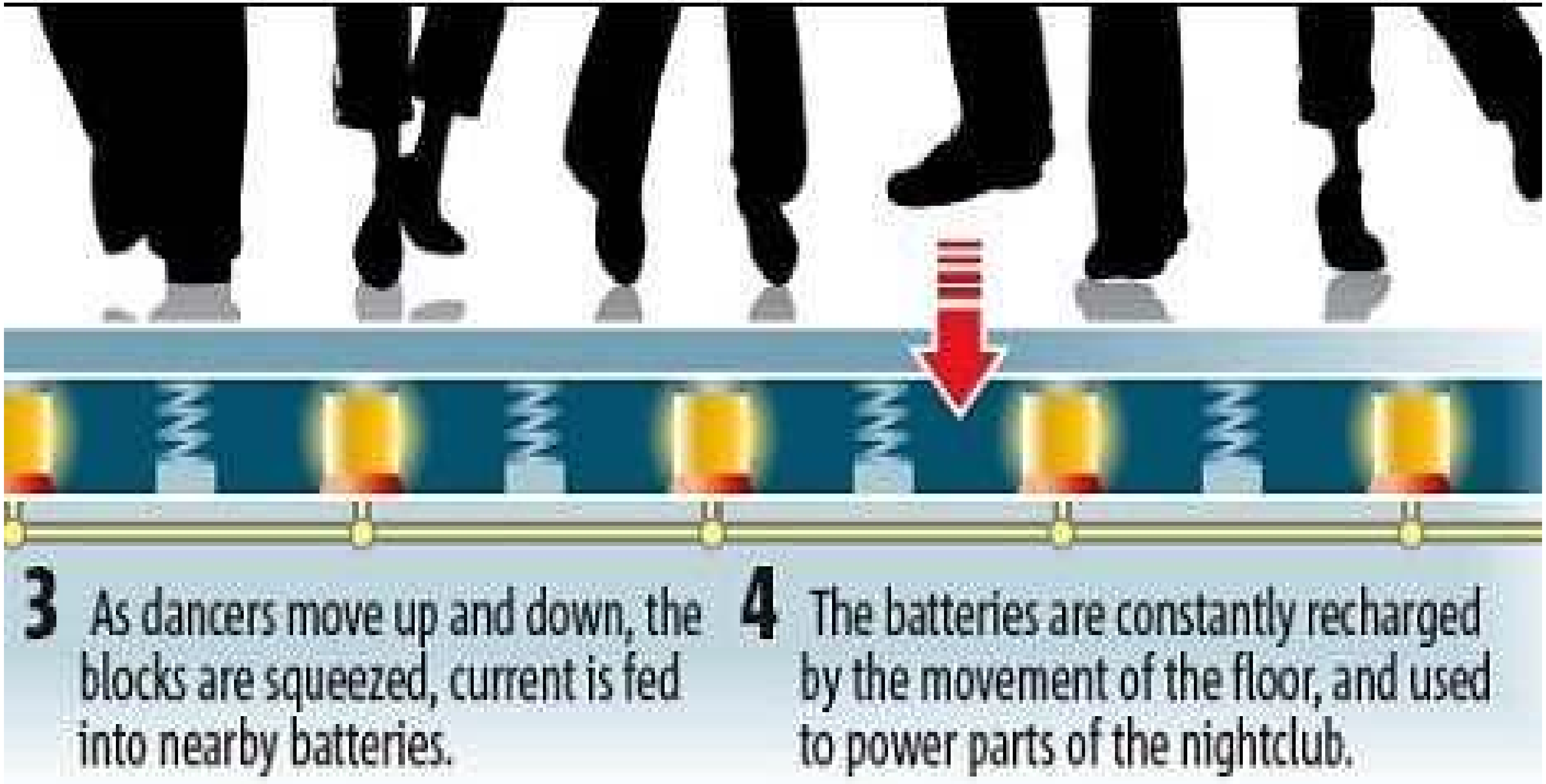
Summary

- 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.

DISCO



Dancefloor Power
How it works
Dancefloor
1
1 Nightclub is fitted with a 'bouncing' floor made of springs and a series of power generating blocks.
2 The blocks made from crystals produce small electrical current when squashed, a process known as piezoelectricity.


3 As dancers move up and down, the blocks are squeezed, current is fed into nearby batteries.
4 The batteries are constantly recharged by the movement of the floor, and used to power parts of the nightclub.