CE323/CE860 Advanced Embedded Systems Design

Dongbing Gu

School of Computer Science and Electronic Engineering
University of Essex
UK

Spring 2018

Outline

Section 1

Introduction to the Course

Course Structure

- Lectures:
  - 20 lectures, 2 lectures per week,
  - Room: TC2.12+TC2.13
  - Time: 12:00-14:00, Monday
  - Revision lecture 12:00-14:00 Monday in Week 30 (23/04/2018), Room TC2.12+TC2.13
- Labs:
  - 11:00-13:00 Thursday (week 16-25)
  - 13:00-15:00 Thursday (week 16-25)
  - Two hours per lab
  - mbed and extension board in Laboratory 3.511 (Lab 7)
  - GTA: Mr. Ruihao Li

Learning Outcomes

The aim of this module is to provide the practical and theoretical skills needed to understand the design of embedded systems. After completing this module, students should be able to:

- Understand the embedded systems design process.
- Define a specification for an advanced embedded system.
- Understand the design process of an embedded system based on a specification.
- Discuss methods for implementing reliable embedded systems to solve various problems.
- Compare and contrast different options for the realisation of advanced embedded systems and their suitability for their application domain.

Assessment

- CE323 Assignment 1
  - Report (8%): **Week 20**, 16 February 2018, Electronic.
- CE323 Assignment 2
  - Report (17%): **Week 25**, 23 March 2018, Electronic.
- CE860 Assignment 1
  - Report (15%): **Week 20**, 16 February 2018, Electronic.
- CE860 Assignment 2
  - Report (20%): **Week 25**, 23 March 2018, Electronic.
- Exam
  - 2 hours.

Recommended Reading

- Recommended Reading:
  - W. WOLF, Computers as Components: Principles of Embedded Computing System Design, Morgan Kaufmann, 2005, ISBN: 0123694590.
  - Rob Toulson and Tim Wilmshurst, Fast and Effective Embedded Systems Design: Applying the ARM mbed, Elsevier, 2012, ISBN: 9780080977683
- Background Reading:
  - PETER MARWEDEL, Embedded System Design, Springer, 2006
  - WALLS, C., Embedded Software: The Works, Elsevier, 2005, ISBN: 0750679549
  - NOERGAARD, T., Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers, Elsevier, 2005, ISBN: 0750677929

Further Details

- There are various relevant books in the library:
  - Books on embedded systems and control applications.
  - Books on C programming.
- mbed references
  - mbed LPC1768 board
  - mbed NXP LPC1768 Microcontroller Flyer
  - mbed NXP LPC1768 Schematic
  - mbed NXP LPC1768 Getting Started
  - NXP LPC1768 Homepage
  - mbed Handbook
  - Course Notes

Lecture Plan

- **Embedded System Design Process**
- **Requirement and Specification**
- **Hardware: Processors, Memories, I/O, Analogue Signals**
- **Networking: I2C, SPI, RS232, RS485, Modbus, CAN, Ethernet**
- **Software: Concurrency, Multitasking, and Operating Systems**

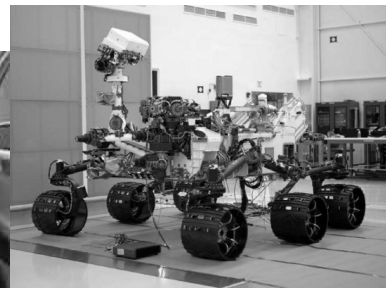
Section 2

Introduction to Embedded Systems

What are Embedded Systems?

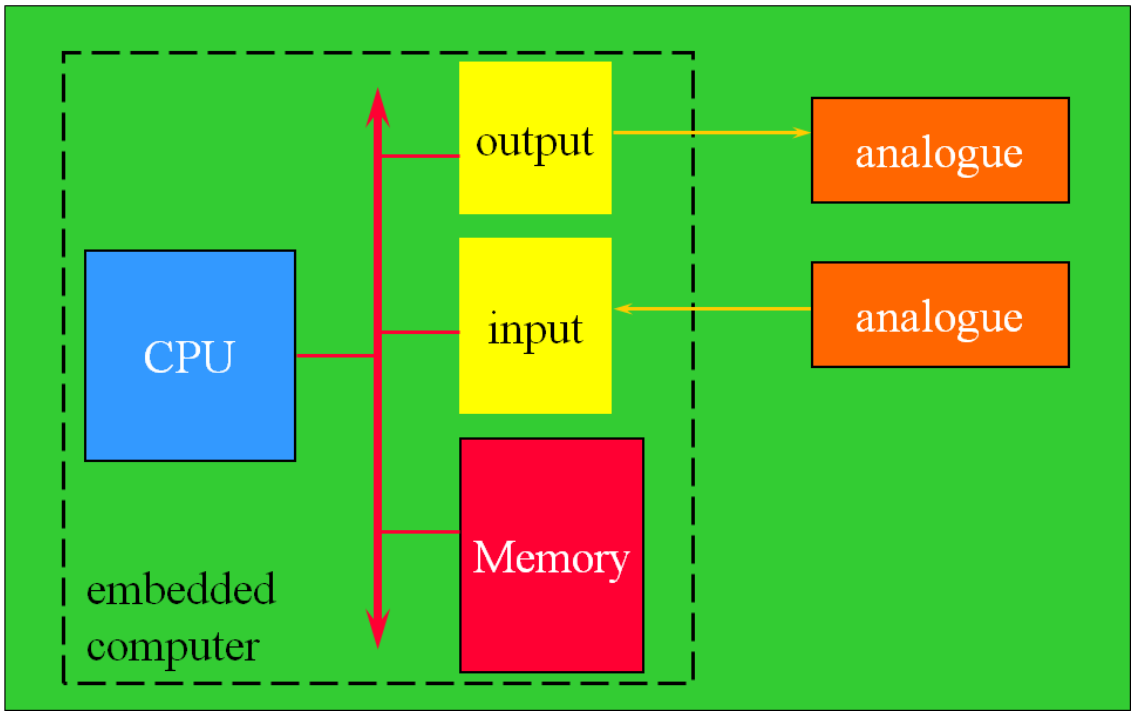
- Any device that includes a programmable computer but is not itself a general-purpose computer.
- Take advantage of application characteristics to optimise the design, don't need all the general-purpose features of a general-purpose computer.
- Account for $> 99\%$ of new microprocessors
  - Consumer electronics
  - Vehicle control systems
  - Medical equipment
  - etc.





Embedding a Computer

- An embedded computer comprises of a microprocessor that serves as the central processing unit (CPU), coupled with input/output (I/O) ports and an addressable memory space.
- I/O ports and interfaces are connections that provide a data path between the CPU and external devices in the analogue or digital world.



Embedding a Computer

- Microcontroller: A system on chip that contains extra support for dealing with the real world
- Analogue to digital and digital to analogue converters
- Embedded networks: SPI, I2C, UART, CAN, Ethernet, etc.
- General-purpose I/O pins
- Lots of interrupt lines
- Low-power sleep modes
- Memories
- On-board volatile and non-volatile RAM
- What else?

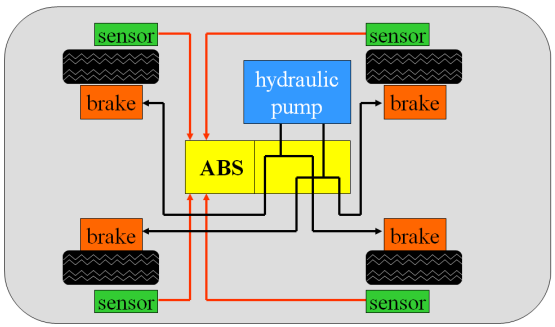
Main kinds of functionality

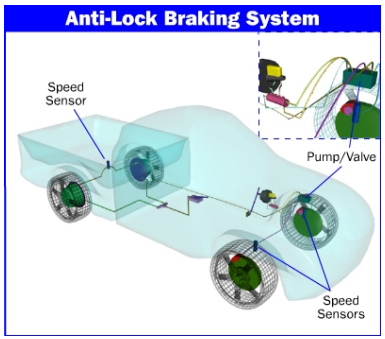
- Digital signal processing
- Open loop and closed loop control
- Wired and wireless networking
- User interfacing
- Storage management

which apply to: mobile phone, home router, cruise control, traffic light?

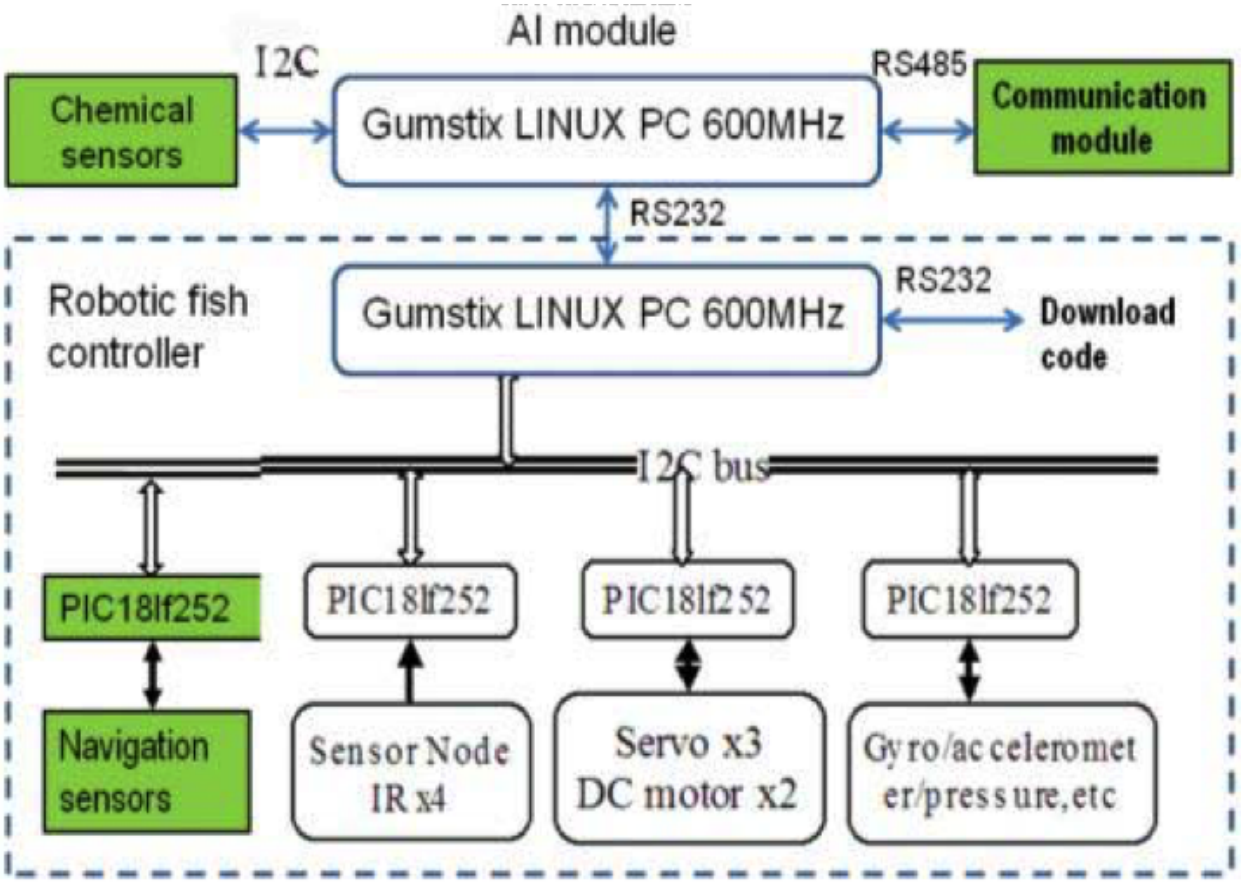
Automotive embedded systems

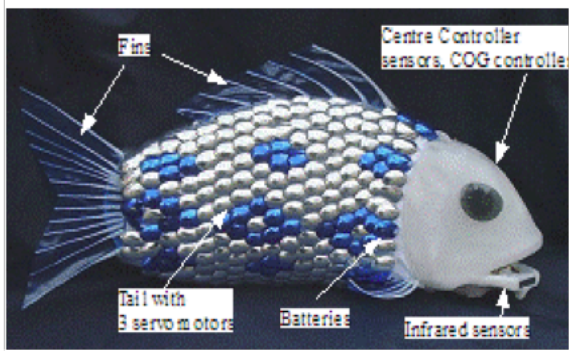
- Today's high-end automobile may have 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.
- BMW 850i Anti-lock brake system (ABS):
  - Main components: a controller, speed sensors, valves (brakes), and a pump.
  - The controller constantly monitors the rotational speed of each wheel, and control the hydraulic pressure to the brake.
  - It allows the wheels to maintain tractive contact with the road surface according to driver inputs while braking, preventing the wheels from locking up (ceasing rotation) and avoiding uncontrolled skidding.



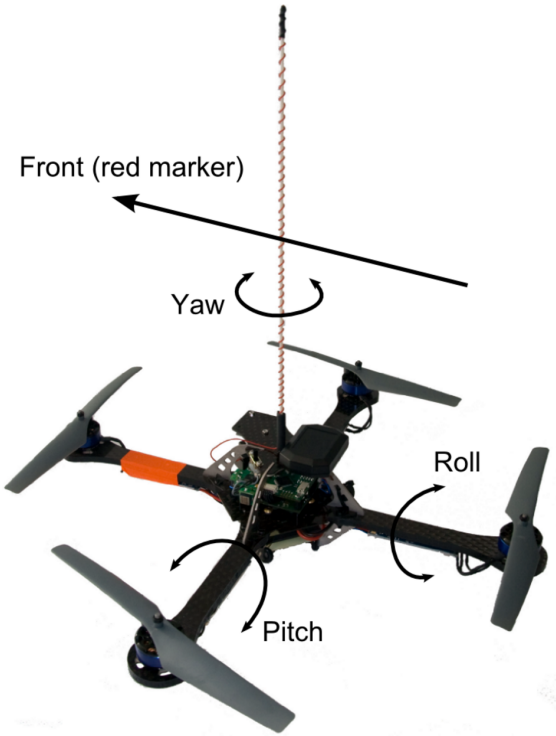


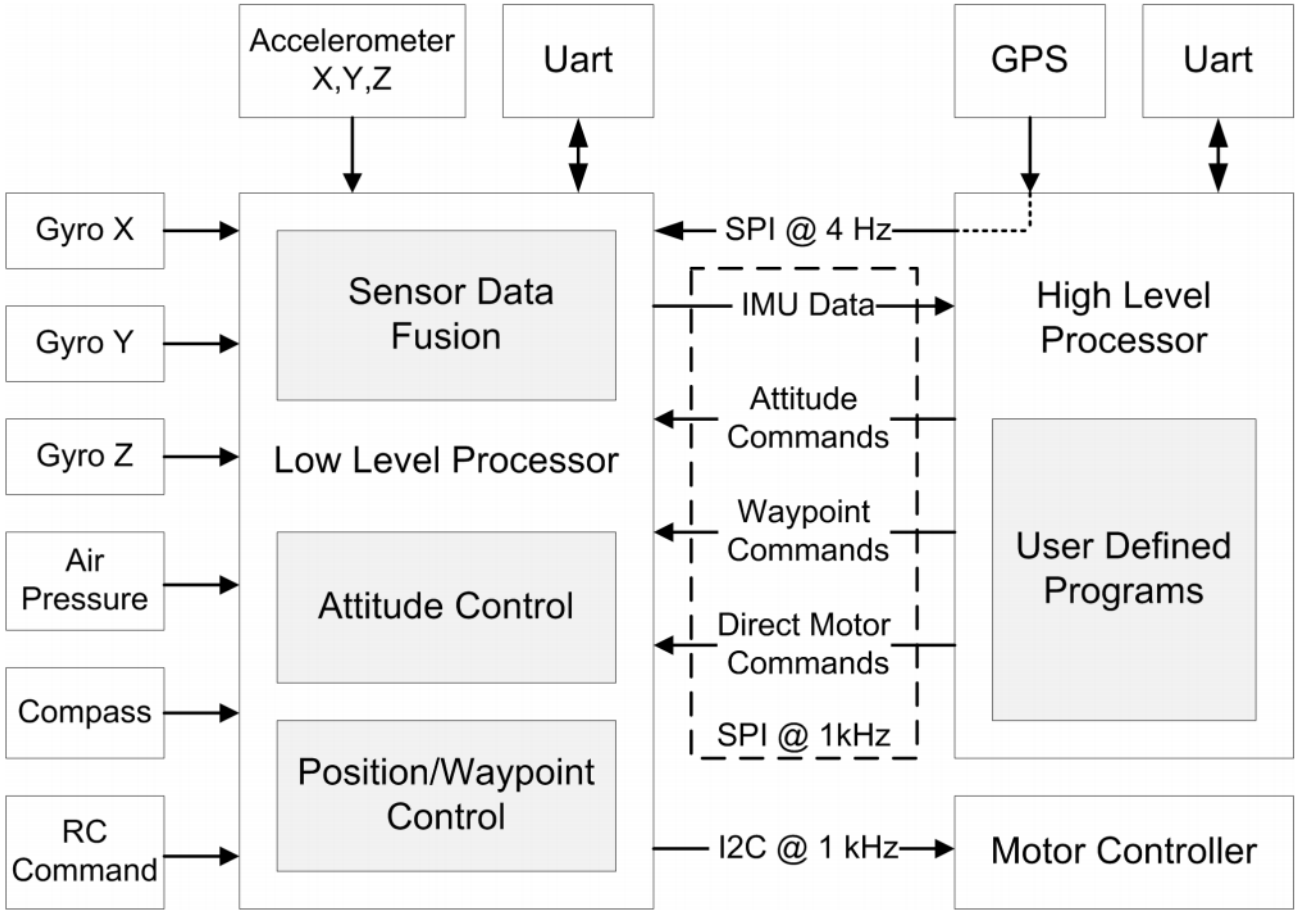
Robotic Fish





Drones


Front (red marker)
Yaw
Roll
Pitch


Accelerometer X,Y,Z
Uart
GPS
Uart
Gyro X
Gyro Y
Gyro Z
Air Pressure
Compass
RC Command
Sensor Data Fusion
Low Level Processor
Attitude Control
Position/Waypoint Control
SPI @ 4 Hz
IMU Data
Attitude Commands
Waypoint Commands
Direct Motor Commands
SPI @ 1kHz
I2C @ 1 kHz
High Level Processor
User Defined Programs
Motor Controller

Section 3

Characteristics and Challenges

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.
  - Automobile engine controller, mobile 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.
- Non-recurring engineering (NRE) cost refers to the one-time cost to research, develop, design and test a new product. When budgeting for a new product, NRE must be considered to analyse if a new product will be profitable. It is becoming very high.
- Power consumption is critical in battery-powered devices.

Tight Deadlines

- Often designed by a small team of designers.
- Often must meet tight deadlines.
  - 6 month market window is common.
  - e.g. Can't miss back-to-school window for a new "scientific calculator".
- Design time has to be reduced!
  - Good design methodologies.
  - Efficient design tools.
  - Reuse of previously designed and verified (hardware and software) blocks.
  - Good designers who understand both software and hardware!

Modern Microprocessors

- Microprocessors use much more logic to implement a function than does custom logic.
- But microprocessors often at least have the following features:
  - Modern microprocessors execute programs very efficiently, with clever utilisation of parallelism to achieve heavily pipelined stages in the CPU;
  - Microprocessor manufacturers spend a great deal of money to make their CPUs run very fast.
  - The use of the latest generation of VLSI fabrication technology can make a huge difference in performance.

Power Saving

- Custom logic is a clear winner for low power devices.
- Modern microprocessors offer features to help control power consumption, such as turning off unused circuits and dynamically trimming clock speeds.
- Software design techniques can help reduce power consumption.
  - Many software power management algorithms achieves this design goal by implementing some form of voltage or frequency scaling in conjunction with hardware support.
  - Memory accesses are a major source of power consumption; Carefully planned memory transactions to avoid reading the same data several times will reduce this.

Section 4

Embedded System Design Process

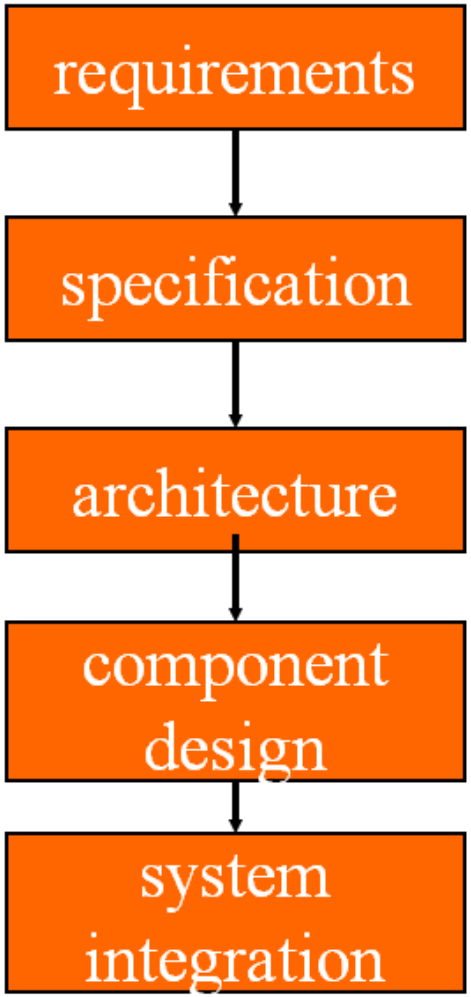
Design Methodologies

- Process for creating a system for a complex systems.
  - everyone has a design process in mind when designing an embedded system;
  - multiple designer team demands a design process.
- Many systems are complex:
  - large specifications;
  - multiple designers;
  - interface to manufacturing.
- Proper processes improve:
  - quality;
  - cost of design and manufacture.

Design Goals

- Functionality and user interface.
- Performance - overall speed, etc.
- Manufacturing cost.
- Power consumption.
- Design cost for a few copies is different with mass market.
- Time-to-market:
  - beat competitors to market;
  - meet marketing window.
- Other requirements (physical size, etc.)

Levels of Abstraction

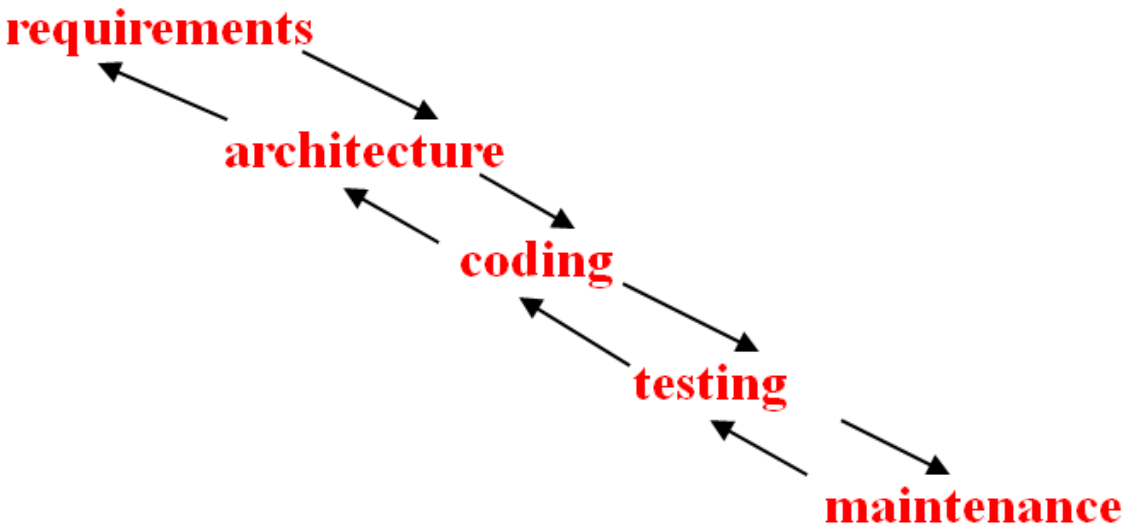


Top-down vs. Bottom-up

- Top-down design:
  - start from most abstract description; work to most detailed.
- Bottom-up design:
  - work from small components to big system.
- Real design uses both techniques.
- May be partially or fully automated, such as using software tools to transform, verify design.

Waterfall model

- Early model for software development (first model):
- Requirements: determine basic characteristics.
- Architecture: decompose the functionality into basic modules (components).
- Coding: implement and integrate.
- Testing: exercise and uncover bugs.
- Maintenance: deploy, fix bugs, upgrade

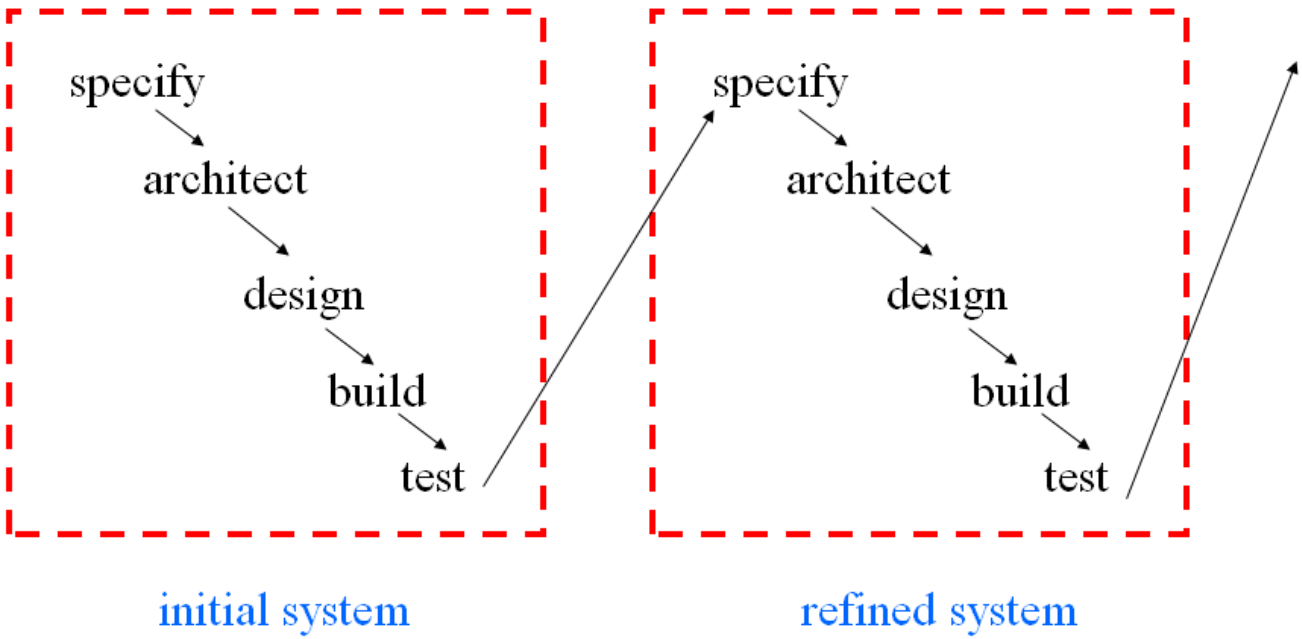


Waterfall model critique

- One-way flow of work and information from higher levels of abstraction to more detailed design.
- limited feedback between levels. Only local feedback - may need iterations between coding and requirements.
- Doesn't integrate top-down and bottom-up design (most designs need bottom up experience).
- Assumes hardware is given.
- sometimes, unrealistic.

Successive refinement model

- In successive refinement model, sub-components within an implementation may be re-designed and replaced without rebuilding the whole system.
- Each implementation is not necessarily a complete system since the goal is to refine sub-components in order to create the optimal final system.

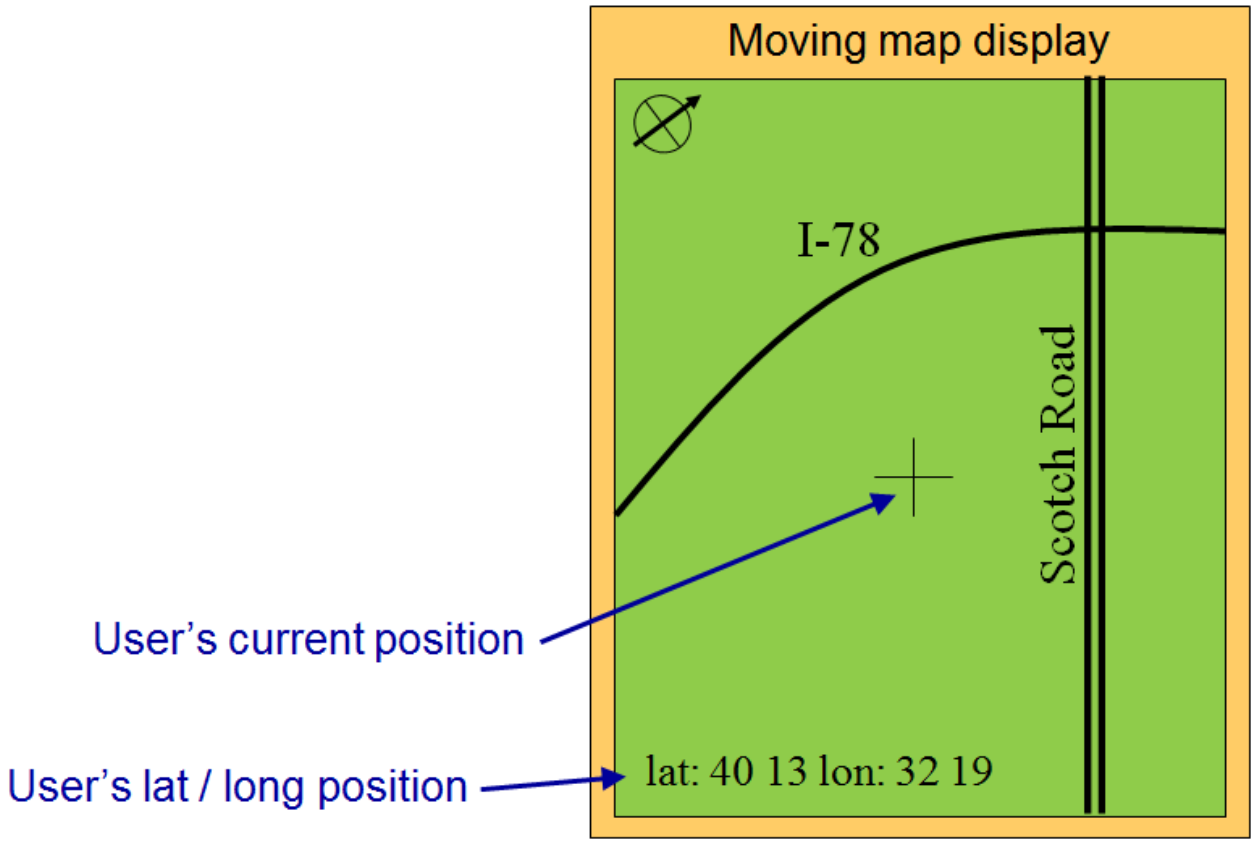


Section 5

Example: GPS Moving Map

GPS Moving Map

- Moving map obtains position (Latitude and Longitude) from GPS, paints map from local database.



Example: GPS Moving Map Requirements

- Functionality:
  - For automotive use. Show roads and other landmarks available.
- User interface:
  - At least 400×600 pixel screen.
  - Three buttons, a menu should pop-up when buttons pressed to allow user's selection.
- Performance:
  - Map should scroll smoothly.
  - No more than 1 sec power-up. verify and display GPS information within 15 seconds.
- Cost: less than £100 shop price - about £40 cost of hardware.
- Physical size/weight: Should fit in dashboard.
- Power consumption: Current draw comparable to CD player.

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 400×600
Functions	receiver; 3 user selectable resolutions; displays current lat/long
Performance	updates screen within 0.25 sec of movement
Manufacture costs	£40
Physical size/weight	no more than 2×6 inches, 12 oz
Power	100mW

Specification

- A more precise description of the system, should provide 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

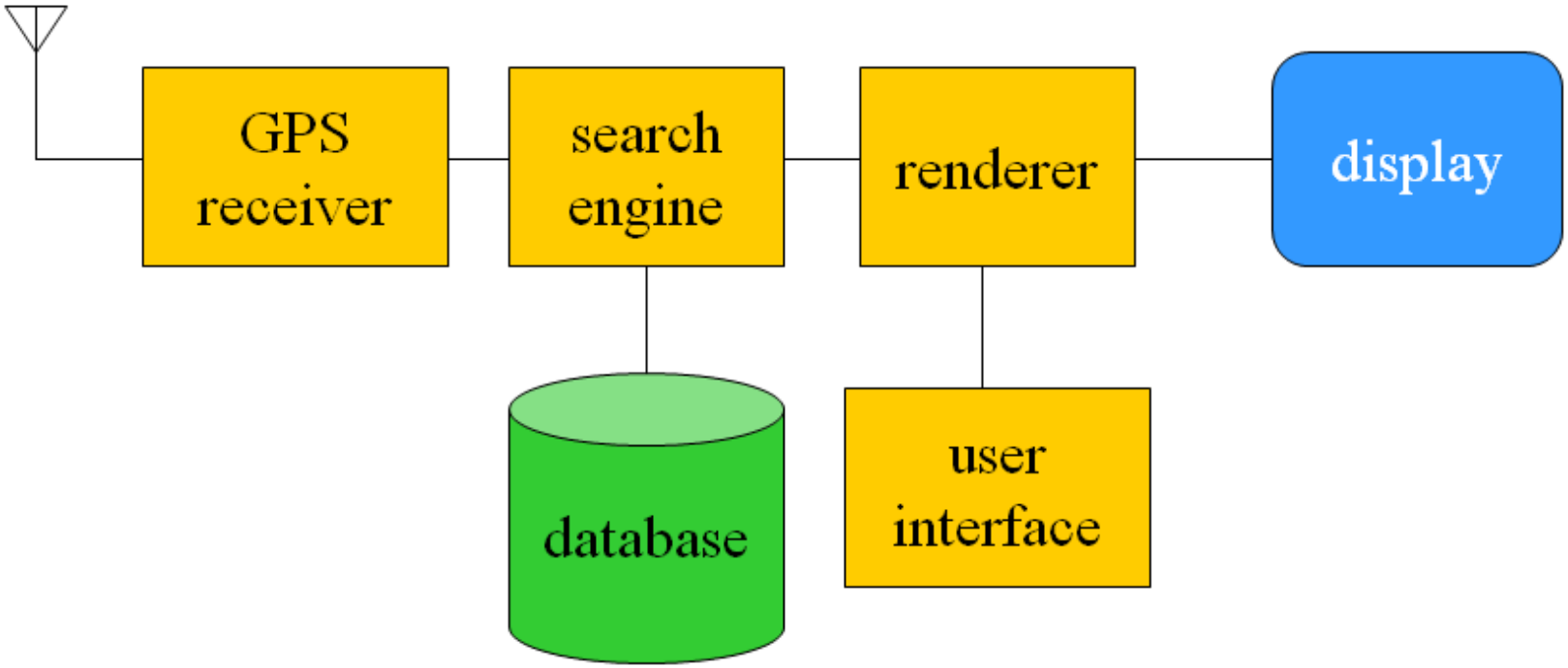
- what is received from GPS;
- map data;
- user interface;
- operations required to satisfy user requests;
- background operations needed to keep the system running.

Architecture Design

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

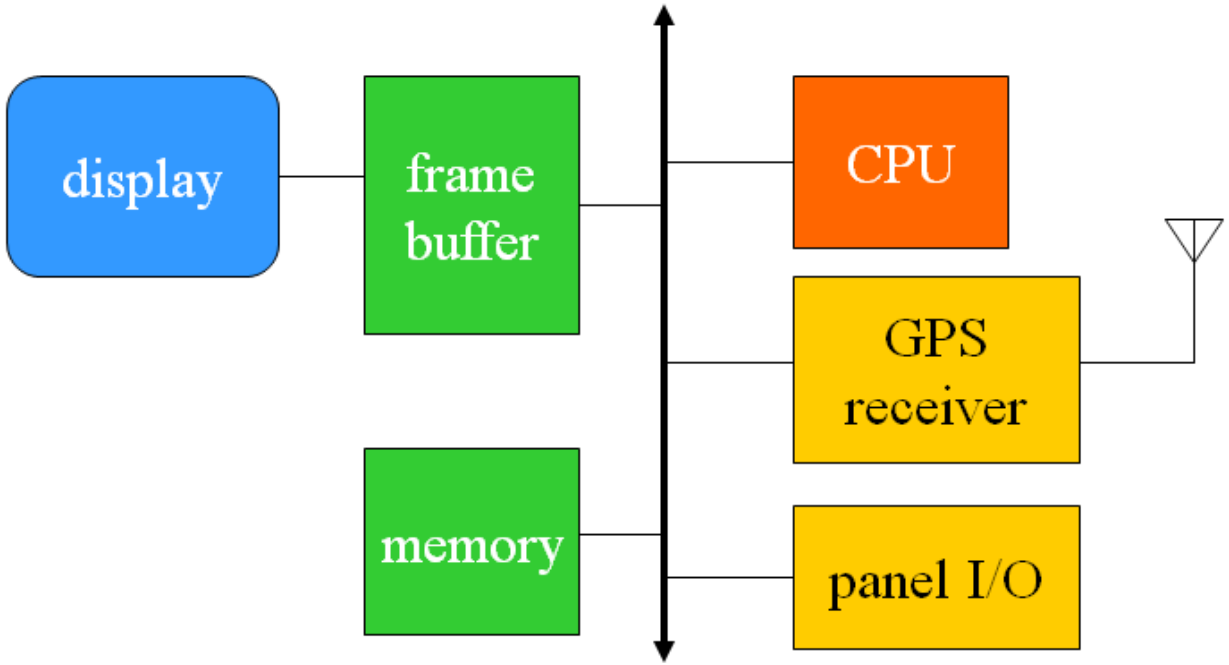
GPS Moving Map Block Diagram

- Block diagram: major operations and data flows among them.



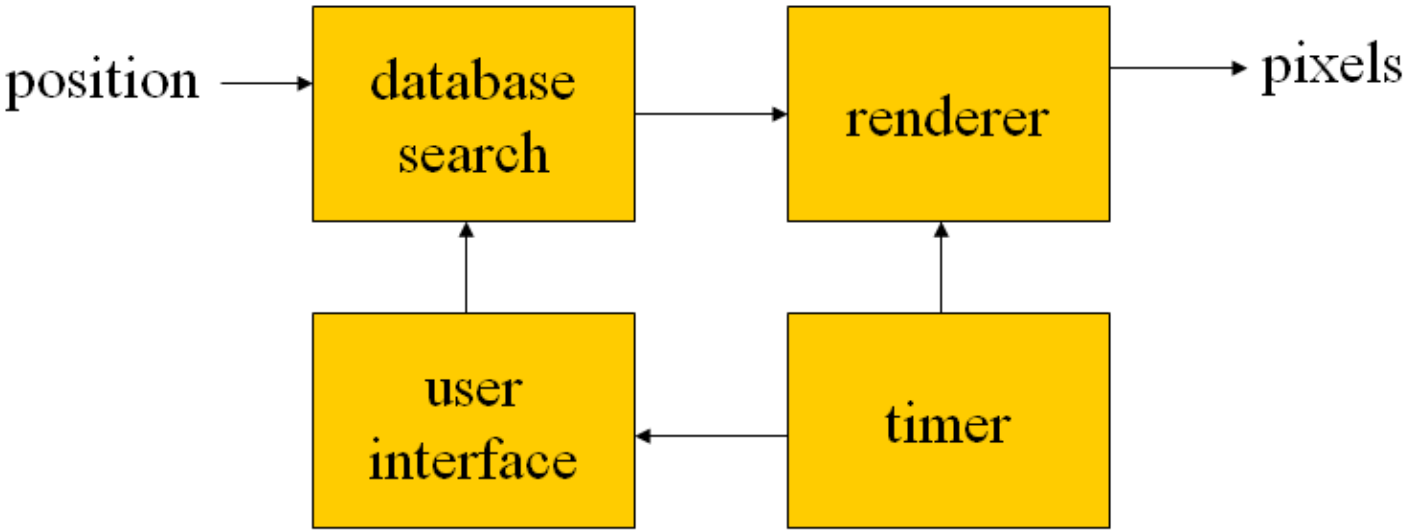
GPS Moving Map Hardware Architecture

- GPS moving map hardware architecture.



GPS Moving Map Software Architecture

- GPS moving map software architecture.



Component Design and System Integration

- Designing hardware and software components.
  - Must spend time designing 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.
- The first step is to produce the requirements of a system.