Characteristics and Challenges of Embedded Computing Systems

1. Characteristics of Embedded Systems

Embedded computing is in many ways more demanding than programming for PCs or workstations. Functionality matters in both, but embedded applications must also meet many other constraints.

1.1 Sophisticated Functionality

- **Complex algorithms:** the microprocessor may run very sophisticated operations. Example: an automobile engine controller performs complicated filtering to optimise performance while minimising pollution and fuel use.
- **User interface:** microprocessors often control complex interfaces with multiple menus and many options. Example: moving maps in GPS navigation.

1.2 Deadline Constraints

- **Real time:** if the data is not ready by a certain deadline, the system breaks. A missed deadline can be unsafe and even endanger lives; in other cases it only creates unhappy customers (e.g. scrambled pages in a printer).
- **Multirate:** several real-time activities run at the same time, some at slow rates and others at high rates. Example: audio and video in a multimedia stream run at very different rates but must stay closely synchronised; missing a deadline on either spoils the whole presentation.

1.3 Cost Constraints

- **Manufacturing cost:** total cost of building the system; determined by the type of microprocessor, the amount of memory and the types of I/O devices.
- **Power and energy:** power consumption affects hardware cost (a larger power supply may be needed). Energy consumption affects battery life and heat dissipation, which matters even in desktop applications.

1.4 Small Teams and Tight Deadlines

- Most embedded systems are designed by small teams on tight deadlines; because a few people can build microprocessor-based systems, management assumes all of them can be built by small teams.
- Tight deadlines are a fact of life in an internationally competitive environment.
- Embedded software still makes sense: hardware and software can be debugged somewhat independently and design revisions can be made much more quickly.

1.5 Summary of Characteristics

Category	Characteristic	Meaning
Functionality	Complex algorithms	Sophisticated processing (e.g. engine control)
Functionality	User interface	Menus, moving maps (GPS)
Deadlines	Real time	Missing a deadline breaks the system

Category	Characteristic	Meaning
Deadlines	Multirate	Slow and fast activities together, kept synchronised
Cost	Manufacturing cost	Processor type, memory size, I/O devices
Cost	Power and energy	Power supply size, battery life, heat

2. Challenges in Embedded Computing System Design

External constraints are one important source of difficulty in embedded system design.

2.1 Design Questions

- **How much hardware do we need?** We control the processor type, memory amount, peripherals and more. Too little hardware and the system misses deadlines; too much and it becomes too expensive.
- **How do we meet deadlines?** The brute-force way is to speed up the hardware, which raises cost. Raising the CPU clock may not help enough because speed can be limited by the memory system.
- **How do we minimise power consumption?** Important in battery-powered and (for heat) non-battery systems. Running slower saves power but can miss deadlines, so only the non-critical parts should be slowed down while performance goals are still met.
- **How do we design for upgradeability?** The same hardware platform may serve several product generations; features are added by changing software. The machine must provide the performance needed by software that has not yet been written.
- **Does it really work?** Reliability matters for every product and especially for safety-critical systems. Waiting until a running system exists to remove bugs is too late: bugs are not all found, are too expensive to fix and take too long.

Note: If workstation programming is like assembling a machine on a bench, embedded system design is often more like working on a car – cramped, delicate and difficult.

2.2 Difficulties from the Nature of Embedded Machines

- **Complex testing:** exercising an embedded system is harder than typing in data; a real machine may have to run to generate proper data. Timing of data is often important, so testing cannot be separated from the machine in which the computer is embedded.
- **Limited observability and controllability:** usually no keyboard or screen. We may have to watch electrical signals on the microprocessor bus, and in real-time applications the system cannot easily be stopped to look inside.
- **Restricted development environments:** tools are more limited than on PCs. Code is compiled on one machine (a PC), downloaded to the embedded system, and debugged with programs on the PC that look inside the embedded system.

2.3 Summary of Challenges

Challenge	Key issue
Amount of hardware	Balance deadlines against manufacturing cost
Meeting deadlines	Faster hardware costs more; memory may be the limit
Power consumption	Slow down only non-critical parts

Challenge	Key issue
Upgradeability	Platform must support future software
Reliability	Bugs must be prevented early, not only fixed late
Complex testing	Needs the real machine and real timing
Observability / controllability	No screen or keyboard; signals must be probed
Development environment	Cross-compile on PC, download, debug remotely