Degrees of difficulty in programming:

- program “runs”
- program satisfies tests
- program is verified
- program is publishable
- somebody wants to use it
Challenges and Difficulties

- Concurrent processes
- Real-time constraints
- Economic considerations
- Involving sensors and actuators
- Low power consumption
- Hostile environments
- Reliability and liability

Mastering Real-Time Programming

- Axiomatic theory
- Discipline of design
- Programming language
- Talent & experience
- Separation of concerns
- Isolation of RT-critical parts
- Urge to reduce complexity
Measures toward simplification

Build custom-tailored hardware
1 StrongARM replacing 2 80486
Remove complex RT-OS
Program in HLL Oberon
involved building new compiler
Compiler built on simple principles
no use of compiler building tools
simple, predictable code has priority
over sophisticated optimization
Rules of thumb

Interrupting tasks steal time from interrupted tasks $\rightarrow$ Avoid interrupts
Caches and pipelines cause unpredictable time requirements $\rightarrow$
Avoid processors with caches
Be skeptical towards complicated tools
Reliability is of foremost concern
There is no Reset Button!

Change in design process

Hardware:
Design in terms of gates, registers $\rightarrow$
Search through data books to find the biggest, most suitable components

Software:
Programming in terms of assignments, iterations and conditions $\rightarrow$
Search through data books to find the biggest, most suitable components
The End of Computer Science?

Is the era of “great inventions” over?
Is this the era of technological perfection?
Are speed and sophistication the only goals?
Are there any missed challenges?
Has Human Interface Design failed?
Is the customer king or victim?
Simplicity theory replaces complexity theory?