

Embedded Real-Time Systems: An Introduction

Raj Rajkumar
Lecture #2

Goals of the Course

High-Level Goals

1. Understand the scientific principles and concepts behind embedded real-time systems, and
2. Obtain hands-on experience in programming embedded real-time systems.

By the end of the course, you must be able to

- Understand the "big ideas" in embedded real-time systems
- Obtain direct hands-on experience on both hardware and software elements commonly used in embedded real-time system design
- Understand basic real-time resource management theory
- Understand the basics of embedded real-time system application concepts such as signal processing and feedback control
- Understand, and be able to discuss and communicate intelligently about
 - I/O and device driver interfaces to embedded processors with networks and multimedia cards
 - OS primitives for concurrency, timeouts, scheduling, communication and synchronization

The Big Ideas

- HW/SW Boundary
- Non-processor-centric view of architecture
- Bowels of the operating system
 - specifically, the *lower* half of the OS
 - Concurrency
- Real-time scheduling
- Analyzability
 - how do you “know” that your drive-by-wire system will function correctly?
- Application-level techniques
 - signal processing, control theory

What *are* Embedded Systems anyway?

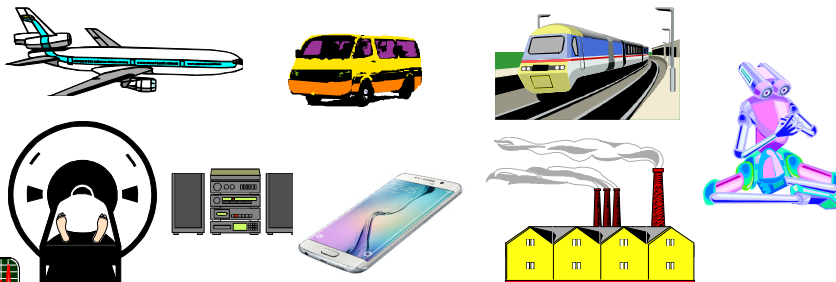
Based on Lecture by Prof. Koopman

Embedded Systems: An Introduction

- What is an embedded system?
 - More than just a computer
- What makes embedded systems different?
 - Real-time operation
 - *Many* sets of constraints on designs
 - size
 - cost
 - time
 - reliability
 - safety
 - energy
 - security
- What embedded system designers need to know?
 - The “big” picture
 - Skills required to be an “expert” in this area

What is an Embedded System?

- Computer purchased as part of some *other/bigger* piece of equipment
 - Typically dedicated software (may be user-customizable)
 - Often replaces previously electromechanical components
 - Often no “real” keyboard
 - Often limited display or no general-purpose display device
- But, every system is unique: there are always exceptions



CPU: An All Too Common View of Computing

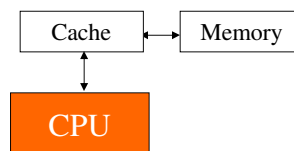
- Measured by:
 - Performance



CPU

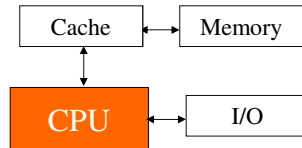
An Advanced Computer Engineer's View

- Measured by: Performance
 - Compilers matter too...



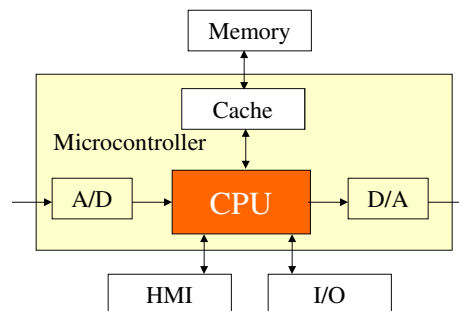
An Enlightened Computer Engineer's View

- Measured by: Performance, Cost
 - Compilers & OS matter



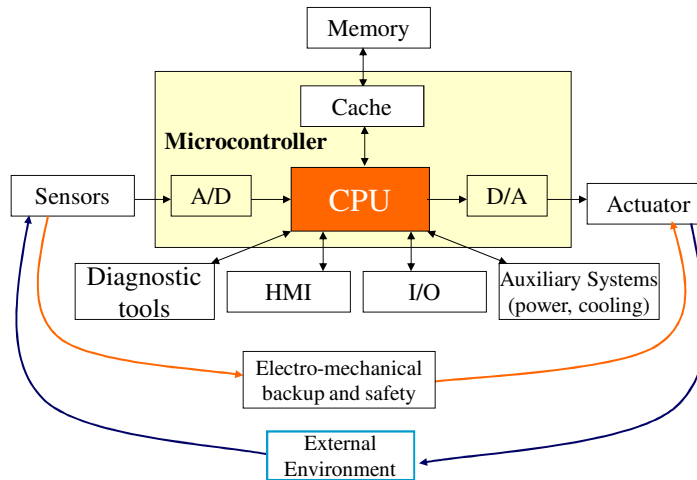
An Embedded Computer Designer's View

- Measured by: Cost, I/O connections, Memory Size, Performance



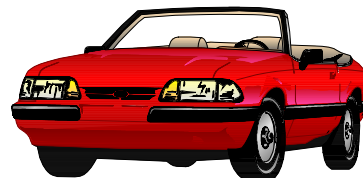
An Embedded Control System Designer's View

- Measured by:
 - Cost, Time-to-market, Cost, Functionality, Cost & Cost.



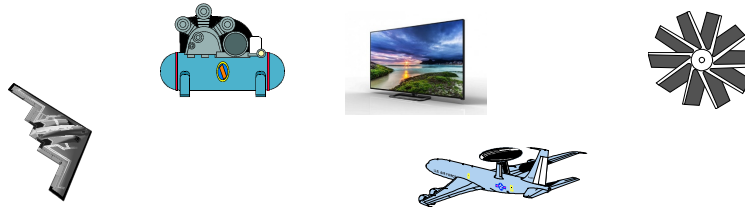
A Customer View

- Reduced Cost
- Increased Functionality
- Improved Performance
- Increased Overall Dependability



Some Embedded System Examples

- Pocket remote control RF transmitter
 - 100 KIPS, water/crushproof, fits in pocket, 5-year battery life
 - Software handcrafted for small size (less than 1 KB)
- Industrial equipment controller (e.g., elevator; jet engine)
 - 110 MIPS for 1 to 10 CPUs, 1 8 MB memory
 - Safetycritical software; realtime control loops
- Military signal processing (e.g., Radar/Sonar)
 - 1 GFLOPS, 1 GB/sec I/O, 32 MB memory
 - Software handcrafted for extremely high performance



Embedded Computers *Rule* the Marketplace

- ~100 Million PCs vs. ~5 Billion Embedded CPUs/microcontrollers annually
 - Embedded market growing; PC market *mostly* saturated
- Smartphones in gray area

Why Are Embedded Systems Different?

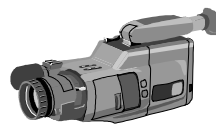
Four General Categories of Embedded Systems

- **General Computing**
 - Applications *similar* to desktop computing, but in an embedded package
 - Video games, set-top boxes, wearable computers, automatic tellers
- **Control Systems**
 - Closed-loop feedback control of real-time system
 - Vehicle engines, chemical processes, nuclear power, flight control
- **Signal Processing**
 - Computations involving large data streams
 - Radar, Sonar, video compression
- **Communication & Networking**
 - Switching and information transmission
 - Telephone system, Internet connectivity



Types of Embedded System Functions

- **Control Laws**
 - PID control
 - Fuzzy logic, ...
- **Sequencing logic**
 - Finite state machines
 - Switching modes between control laws
- **Signal processing**
 - Multimedia data compression
 - Digital filtering
- **Application-specific interfacing**
 - Buttons, bells, lights,...
 - Highspeed I/O
- **Fault response**
 - Detection & reconfiguration
 - Diagnosis



Distinctive Embedded System Attributes

- **Reactive:** computations occur in response to external events
 - Periodic events (e.g., rotating machinery and control loops)
 - Aperiodic events (e.g., button closures)
- **Real-Time: timing correctness is part of system correctness**
 - **“Hard” real-time**
 - Absolute deadline, beyond which answer is useless
 - May include minimum time as well as maximum time
 - **“Soft” real-time**
 - Missing a deadline is not catastrophic
 - Utility of answer degrades with time difference from deadline
 - **Example:**
 - a train is entering an urban area...
 - the railway gate in the city allows automotive traffic to go over the tracks
 - when should the railway gate close?



In general,



Real Time != “Real Fast”
18-648: Embedded Real-Time Systems

Carnegie Mellon

Typical Embedded System Constraints

- **Small Size, Low Weight (SWAP)**
 - Handheld electronics
 - Transportation applications weight costs money
- **Low Power**
 - Battery power for 8+ hours (laptops often last only 2 hours)
 - Limited cooling may limit power even if AC power available
- **Harsh environment**
 - Heat, vibration, shock
 - Power fluctuations, RF interference, lightning
 - Water, corrosion, physical abuse
- **Safety-critical operation**
 - Must function correctly
 - Must not function incorrectly
- **Extreme cost sensitivity**
 - \$.05 adds up over 1,000,000 units



18-648: Embedded Real-Time Systems

Carnegie Mellon

Embedded System Design World View

A complex set of tradeoffs:

- Optimize for *more than just speed*
- Consider *more than just the computer*
- Take into account *more than just initial product design*

Multi-Discipline



Multi-Objective



Multi-Phase

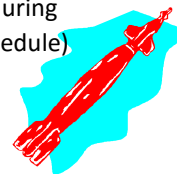
- Electronic Hardware
- Software
- Mechanical Hardware
- Control Algorithms
- Humans
- Society/Institutions

- Dependability
- Affordability
- Safety
- Security
- Scalability
- Timeliness

- Requirements
- Design
- Manufacturing
- Deployment
- Logistics
- Retirement

Mission-Critical Applications Require Robustness

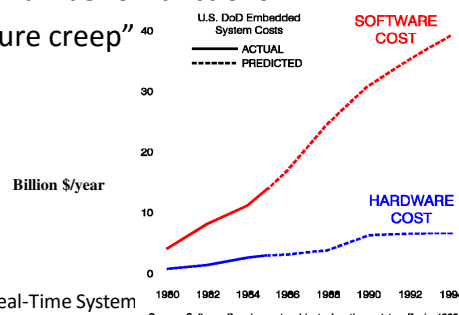
- **Loss of Ariane inaugural flight in June, 1996**
 - Lost a \$400 million scientific payload (the rocket was *extra*)
- **Efforts to reduce system costs led to the failure**
 - Reuse of Inertial Reference System software from Ariane 4
 - Improperly handled exception caused by variable overflow during new flight profile (that wasn't simulated because of cost/schedule)
 - 64-bit float converted to 16-bit int assumed not to overflow
 - Exception caused dual hardware shutdown (software doesn't fail!)
- **What really happened?**
 - The narrow view: it was a software bug fix it
 - The broad view: the loss was caused by a lack of system robustness in an exceptional (unanticipated) situation



Many embedded systems must be robust

Software Drives Designs

- Hardware is mostly a recurring cost
 - Cost proportional to number of units manufactured
- Software is a “one-time” non-recurring engineering design cost (NRE)
 - Paid for “only once”
 - But bug fixes may be expensive, or impossible
 - Cost is related to complexity & number of functions
 - Market pressures lead to “feature creep”
- **Software Is NOT free!!!!**



Life-Cycle Concerns Figure Prominently

- “Let's use a CAD system to re-synthesize designs for cost optimization”
 - Automatically use whatever components are cheap that month
 - Would permit quick responses to bids for new variants
 - Track record of working fine for PC motherboards
- Why wouldn't it work for an automotive application?
 - Embedded systems had more analog than digital mostly digital synthesis tools
 - Cost of recertification for safety, FCC, warranty repair rate
 - Design optimized for running power, not idle power
 - Car batteries must last a month in a parking lot
 - Parts cost didn't take into account lifecycle concerns
 - Price breaks for large quantities
 - Inventory, spares, end-of-life buy costs
 - Tools didn't put designs on a single sheet of paper
 - Archive system: paper-based how else do you read
 - 20-year old files?

Embedded System Designer Skill Set

- Appreciation for multi-disciplinary nature of design
 - Both hardware & software skills
 - Understanding of engineering beyond digital logic
 - Ability to take a project from specification through production
- Communication & teamwork skills
 - Work with other disciplines, manufacturing, marketing
 - Work with customers to understand the real problem being solved
 - Make a good presentation; even better write "trade rag" articles
- And, by the way, technical skills too...
 - *Low-level*: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
 - *High-level*: Object-oriented Design, C/C++, Real Time Operating Systems
 - *Meta-level*: Creative solutions to highly constrained problems
 - *Likely* : Unified Modeling Language, embedded networks
 - *Possibly*: Java, Android, iOS, Windows Mobile

Real-Time Systems



- The correctness of a real-time system is comprised of both logical correctness and timeliness of results

Application Domains (Example Applications)

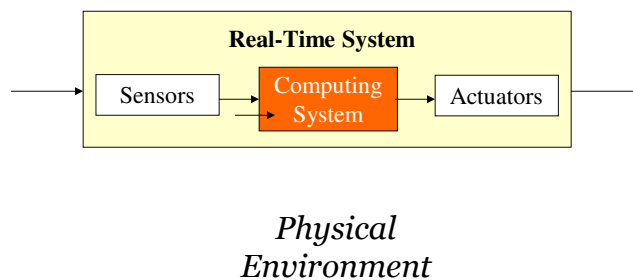
- Process Control (Chemical Plants)
- Factories (Automotive Assembly Plants)
- Supervisory Control and Data Acquisition (Utilities)
- Medical Devices and Healthcare (Medical Robots and Equipment)
- Computer Peripherals (Laserprinters)
- Automotive (Engine controls)
- Telecommunications (Cellphone Infrastructure)
- Aerospace (avionics)
- Internet and Multimedia (Videoconferencing, IP-TV, Skype)
- Consumer Electronics (Mobile phones, TVs, set-top boxes)
- Military systems (missile guidance systems, radar systems)
- Space systems (planetary rovers, rockets, satellites)

Processors in Embedded Real-Time Systems

- Uniprocessors
 - 4-bit microcontrollers
 - 8-bit microcontrollers
 - 16-bit microcontrollers
 - 32-bit microcontrollers
 - 64-bit microcontrollers
 - 128-bit microcontrollers
- Multiple processors
 - Multiple single-board computers on a backplane bus
- Multicore processors
 - Multicore processors
 - Multiple multicore boards on a backplane bus
- Networks of computers
- Networks of multi-processor systems



Generic Model



Typical Characteristics of Real-Time Systems

- Timing constraints
- Embedded nature
- Safety criticality
- Concurrency
- Distributed nature
- Reactive/Feedback system
- Stability
- Exception handling
- Reliability

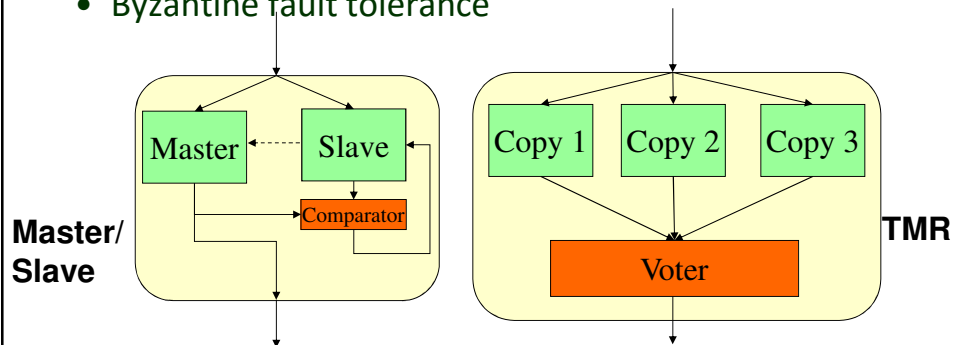
There are always exceptions....

Approaches to Reliability/Fault-Tolerance

- Error Avoidance
- Error detection and removal
- Hardware Fault-Tolerance
- Software fault-tolerance

Hardware Fault-Tolerance

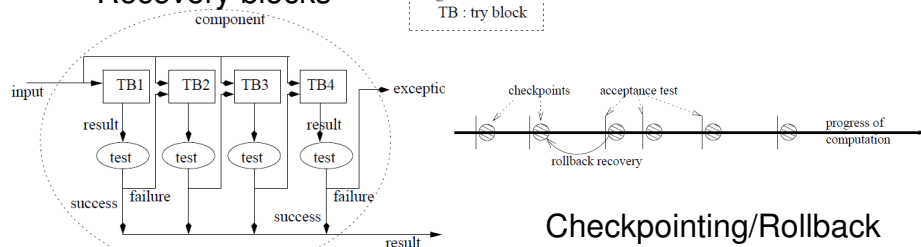
- Built-in Self Testing
- Replication techniques
 - Master-Slave (Primary-Backup Systems)
 - Triple Modular Redundancy (TMR) with Voting
- Byzantine fault tolerance



Software Fault-Tolerance

- *N*-version programming:
 - Multiple versions developed in parallel
- Recovery blocks
 - Try execution, test for correctness, else try another path
- Checkpointing and rollback
- Analytical redundancy
 - Compute same or similar function using other techniques

Recovery blocks



Points in Time

- **Events** represent occurrences in time
- Events can be **periodic** or **aperiodic**
 - Periodic events have a periodic arrival pattern
 - A recurring event without a periodic pattern is called an aperiodic event
- Event sources can be **internal** or **external**
 - Internal events are generated from within the real-time system
 - E.g. timeouts, task completions, lock releases, watchdog timers
 - External events are generated by the environment
 - E.g. a user pressing a button, water leaks, temperature getting too hot (or cold)

Types of Timing Constraints

- **Deadlines (the most common):** action must be taken/completed before a specific instant of time
 - **Hard deadlines:** timing constraints must be met, or else system failure can result
 - E.g. Flight controls
 - **Soft deadlines:** timing constraints must be met when possible; else, the desirability of the system drops but no major damage occurs
 - E.g. internet audio/video conferencing
 - **Firm deadlines:** timing constraints must be met as much as possible with an occasional miss acceptable
 - E.g.
- **Delays**
 - Minimum delay between events

Review

- Embedded system and perspectives
- Real-time systems and characteristics
 - Types of fault-tolerance
 - Events and types
- Coming next: uniprocessor scheduling