General Course Information

- Instructor: Kang G. Shin, 4605 CSE, 763-0391; kgshin@umich.edu
- # of credit hours: 4
- Class meeting time and room:
  - Regular classes: MW 10:30am – noon @1012 EECS
  - Makeup/discussion (as needed): F 10:30 – 11:30am @1012 EECS
- Office hours: MW 9:30 – 10:30am, or by appointment but email is the best way to get hold of me.
- Course homepage: http://www.eecs.umich.edu/courses/eecs571
Important Dates and Class Email

- **Important Dates**
  - Start of class: Sep. 8 (Wed)
  - Study break: Oct. 18-19 (Mon-Tue)
  - One-page project proposal due: Oct. 13 (Wed)
  - Thanksgiving break: 5pm Nov. 24 (Wed)–8am Nov. 29 (Mon)
  - Comprehensive exam: Dec. 1, Wed (tentative)
  - Last day of class: Dec. 13 (Mon)
  - Term project presentations: 6pm-midnight 12/13, 3725 CSE
  - Term project report due: **electronically** by 4pm 12/17 (Fri)

- **Email group**: Subscribe to the mail list by sending email to eecs571-request@eecs with “subscribe” in the Subject field. You may use this email group (eecs571@eecs) only for the class.
Course Materials

- Copies of ``Real-Time Systems,'' Krishna and Shin, McGraw-Hill, 1997 will be made available at Dollar Bill. Errata is maintained on the course URL: http://www.eecs.umich.edu/courses/eecs571/book_correction.pdf and typos and other errors should be reported to me or rtbook@tikva.ecs.umass.edu.


- Four key sources of reading are:
  - ACM Transactions on Embedded Sysems (2002–)

- University Digital Library (http://www.ieeeexplore.org and http://www.acm.org)
Pre-requisites and Grading Policy

- **Pre-requisites**: EECS 482 or EECS 470, or basic knowledge in system software and computer architecture is required, or instructor's approval.

- **Grading Weights**
  - Bi-weekly homeworks: 15%
  - Comprehensive midterm on Dec. 3, 2010: 25%
  - Term project: 55% (presentation 30% and report 25%)
  - Class participation: 5%

- **Collaboration and Regrading Policies**: see the handout or course homepage.

- **Important Information on HWs and Term Projects**: see the handout or course homepage.
General Concepts of Real-Time Embedded Systems

- What's a real-time system and what’s not?
- What’s an embedded system?
- Types of real-time systems
  - Hard real-time systems: definition and examples
  - Soft real-time systems: definition and examples
- What's a deadline and where is it coming from?
  - Law of physics
  - Artificially imposed.
- A task/message/packet may be critical or non-critical, depending on its function and system state.
- Based on invocation/trIGGERing behavior, a task/message/packet is periodic, aperiodic, or sporadic.
- How do we derive message/packet deadlines?
A Typical Real-Time Embedded System

Environment

- Controlled Process
- Sensors
- Store of Jobs
- Real-time Clock
- Trigger Generator
- Execution Unit (Processors, Networks, OS, App SW)
- Human Operators
- Displays

Actuators
Real-time Embedded Systems

- **Embedded system**
  - The software and hardware component that is an essential part of, and inside another system

- **Real-time system**
  - needs timely computation
  - deadlines, jitters, periodicity
  - temporal dependency
Real-time Embedded Systems

- Conventional Dedicated Systems
  - Unique solution (HW/SW/tool) for each application
  - System + domain knowledge

- Hardware (processor, memory, I/O, bus, etc.)
- Software (OS, libraries, application, GUI, etc.)
- Design process (specification, development, testing, etc.)
- Tool chain (analysis, compiler, debugging, integration, etc.)
Embedded Systems

- are everywhere
- How many embedded processors in your home?
  40-50 estimated in 1999.
- What are they?

Hardware (chips) + Software (program)

- CPU processor (ARM, PowerPC, Xscale/SA, 68K)
- Memory (256MB or more)
- Input/output interfaces (parallel and serial ports)
Requirements for RTES

- Environmental – size, power (heat), weight, and radiation-hardened
- Performance – responsive, predictable (fast?)
- Economics – low cost and time-to-market
- Consequence – safety, faulty-tolerance, security
  - DO 178b (avionics)
  - FDA 247 (medical devices)
  - ANS 7.4.3.2 (nuclear power plants)
  - Mil-Std 882d (weapon systems)
- Smaller, cheaper, better, and faster
To write the control software for a smart washer
- initialize
- read keypad or control knob
- read sensors
- take an action

Current system state
- state transition diagram
- external triggers via polling or ISR

If there are multiple triggers and external conditions – single or multiple control loops
Periodic Tasks

- Invoke computation *periodically*
  - Adjust pressure valves at a 20 Hz rate

```
Task initialization
(set up periodic timer interrupts)

wait for the interrupt event

computation

start_time=time()

computation

Sleep(period - (time() - start_time))
```
SW Development for RTES

- Never-ending in a single control loop
- Single execution thread and one address space
- Event- and/or time-driven state transitions
- Small memory footprint (?)

What are missing in the previous example?
- no concurrency (real-world events occur near simultaneously)
- no explicit timing control (add a timer)
- difficult to develop and maintain large embedded systems – verifiable, reusable, and maintainable
Multi-tasking for concurrent events
Machine dependency and portability
Software abstraction, modular design
  - information hiding, OO, separate compilation, reusable
  - a sorting procedure – function, input, output specification
Control timing
Resource constraints and sharing
  - CPU time, stack, memory, and bandwidth
Scheduling
  - Tasks, messages, and I/O
Timing Constraints and Characteristics

- Predicting and controlling timing and events

- Timing relationship: (can you guarantee it?)
  - predictable actions in response to external stimuli
  - deadline (absolute or relative), and jitter

- Instruments play in a band
  - miss a note or timing

- Difficult to control timing
  - all players of an interactive game in Internet see the actions at the same time
  - Sequence, order, and race condition
Timing Constraints and Multi-threading

- Given input $x_1$ at time $t_1$, produce output $y_1$ at time $t_2$
- Non-deterministic operation, time-dependent behavior, and race condition
  - difficult to model, analyze, test, and re-produce.
- Easy to identify the faults and fix them once the failing sequences are reproduced (or observed), but
  - The failures are rooted in the interaction of multiple concurrent operations/threads and are based on timing dependencies
Embedded System Development

- Need a real-time (embedded) operating system?
- Need a development and test environment?
  - Use the host to edit, compile, and build application programs, and configure the target
  - At the target embedded system, use tools to load, execute, debug, and monitor (performance and timing)
**Real-time Operating System (RTOS)**

- **Functions:**
  - task management,
    - scheduling, dispatcher
    - communication (pipe, queue)
    - synchronization (semaphore, event)
  - memory management
  - time management
  - device driver
  - interrupt service

---

**Diagram**

- **External interrupt**
- **Timer interrupt**
- **System calls (trap)**
- **Interrupt dispatch**
- **Interrupt service**
- **Time service & events**
- **Services (create thread, sleep, notify, send, …)**
- **Scheduling & dispatcher**
- **Task execution**
- **Kernel**
Development Environment

- Use the host to
  - edit, compile, build application programs, and configure the target

- At the target embedded system, use tools to
  - load, execute, debug, and monitor (performance and timing)

- The target server manages the interactions with the target
  - communication channel
  - symbol table for the target
Trends in Embedded Systems

- Data from Japan ITRON survey for new embedded systems

![Type of Processors](image1.png)

![Programming Languages](image2.png)
Trends in Embedded Systems

RTOS Used

- 1998-99
- 1999-00
- 2000-01

Difficulties with RTOS

- Inadequate vendor support
- Lack of software components
- Performance and functions do not meet
- Big OS size and resource use
- Differences in OS spec., hard to switch
- Lack of development environment and tools
- Cost is too high
- Other
- Nothing

Lack of engineers familiar with it
Major Topics of RT ES

- Performance measures & task execution time estimation
- Task assignment & scheduling
- Real-time OS and other system software
- Power management for CPU, memory and disk
- Time-sensitive wired and wireless networking
- Security and privacy of embedded systems and devices
- Model-based integration of embedded real-time software
- Formal methods
- Fault-tolerance of embedded real-time systems
- Clock synchronization
- Applications: multimedia, VoIP/VoWLAN, VoD, info and home appliances, medical devices, sensors & actuators, virtual reality, automotive electronics (powertrain controls and infotainment systems, ITS), automated manufacturing, large embedded systems (ships, planes),...