Embedded Systems Programming

Introduction
(Module 1)

Yann-Hang Lee

Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014

Course Syllabus

- **Course Goals:**

  fundamental issues as well as practical development skill in the area of embedded systems programming

  - Design issues of embedded software and the knowledge of development and execution environment on target processors.
  - The functions and the internal structure of device interfaces, drivers, and real-time operating systems.
  - Multi-threaded embedded software in target environment
  - Task scheduling and schedulability analyses.

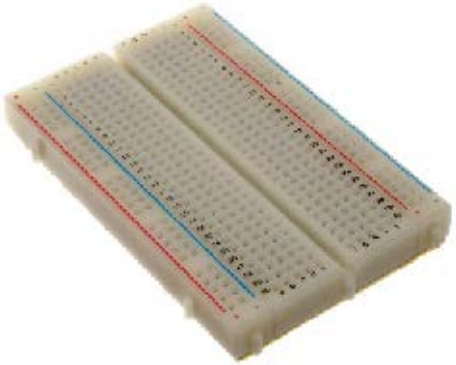
- **Pre-requisites:**
  - Computer organization, data structures, and C/C++ programming
  - Helpful if you have some knowledge of operating systems and computer architecture.

Schedule

Date	Classes (4 20-minutes lectures/day)	Lab and self-study (3 hours/day)
Monday, July 7	Introduction, Linux loadable modules	Exercise: data structures for Linux device drivers
Tuesday, July 8	Linux device driver	Lab: Linux loadable module
Wednesday, July 9	Quark SOC and Galileo architecture	Exercise: Galileo board design and GPIO programming
Thursday, July 10	Linux ISR and device driver	Exercise: user-level I2C programming
Friday, July 11	Thread and kernel synchronization	Lab: I2C-based EEPROM driver
Monday, July 14	Embedded programming	Exercise: setjmp and longjmp
Tuesday, July 15	Embedded programming	Lab: signal and asynchronous control
Wednesday, July 16	Real-time scheduling and analysis	Self-study: course review
Thursday, July 17	Real-time scheduling and analysis	Lab: real-time task management
Friday, July 18	Real-time scheduling and analysis	Final exam

Lab Setup

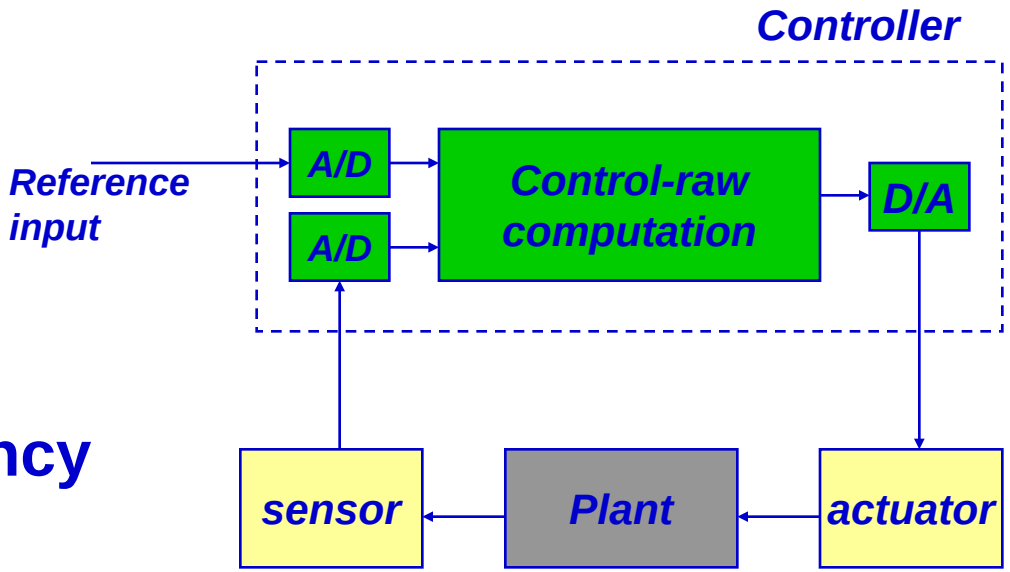
- **OMAP beagleboard XM and trainer board**



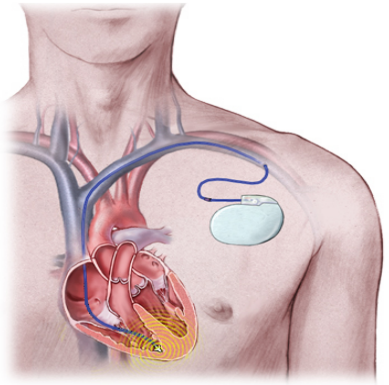
- **Prepare your own micro SD card (and adapter) for booting**
- **Preferred development software**
  - Linux, Eclipse, command lines, GUN tools

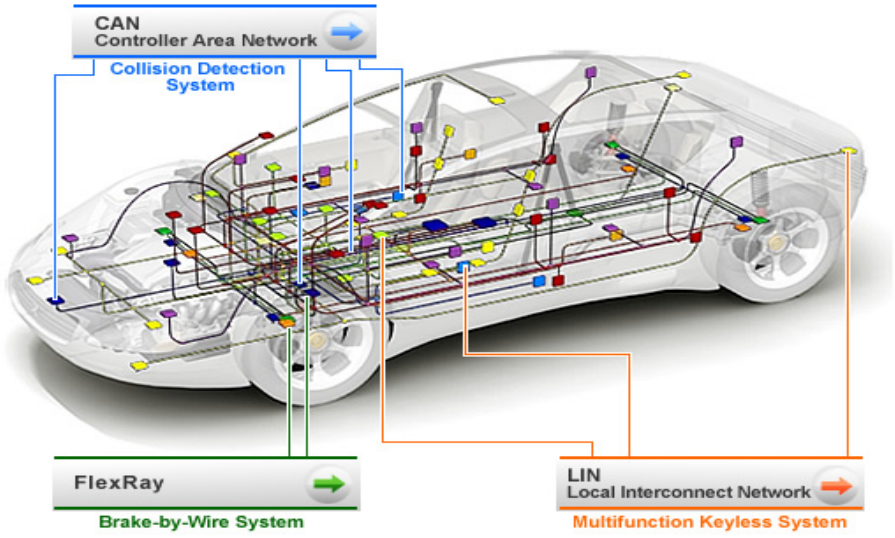
Real-time Embedded Systems

- ***Embedded system***
  - **the software and hardware component that is an essential part of another system**
- ***Real-time system***
  - **provide well-timed computation**
  - **deadlines, jitters, periodicity**
  - **temporal dependency**



Embedded Systems -- Examples



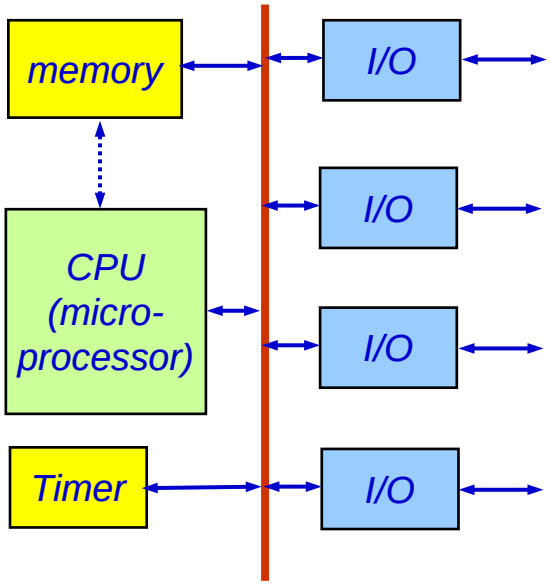


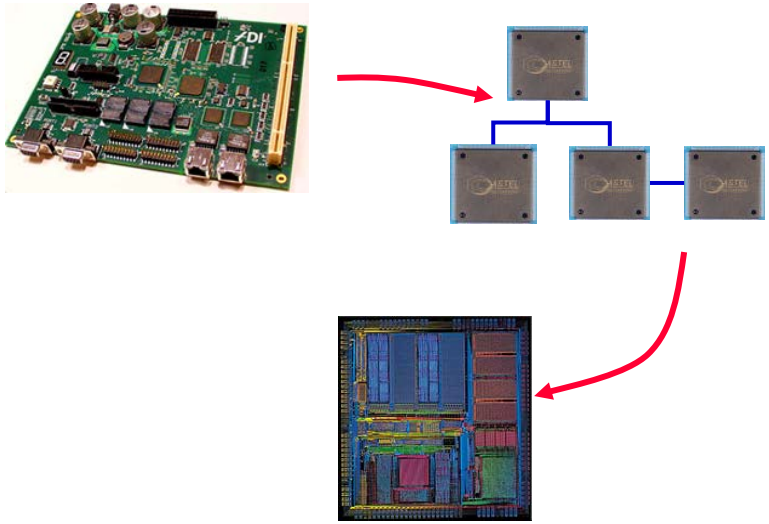
Emerging Embedded Systems

Embedded Systems

- They are everywhere
- What are they?

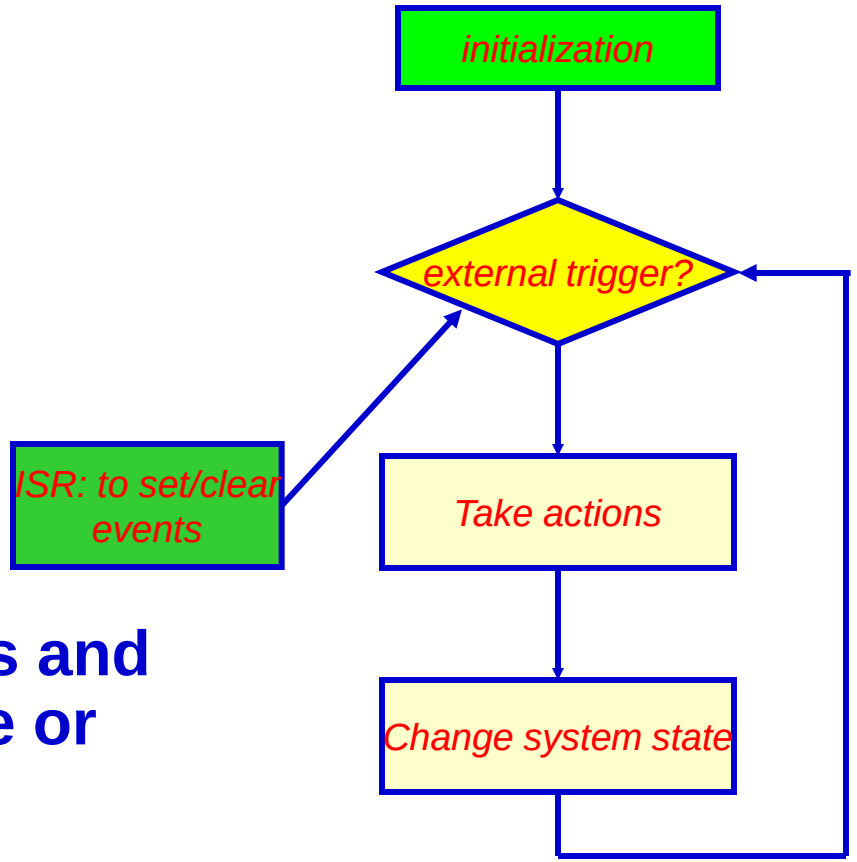
Hardware (chips) + Software (programs) for specific applications





SW Development for RT ES

- **To write the control software for a smart washer**
  - initialize
  - read keypad or control knob
  - read sensors
  - take an action
- **System current state**
  - state transition diagram
  - external triggers via polling or ISR
- **If there are multiple triggers and external conditions – single or multiple control loops**



RT ES vs. General Software

- **Multi-tasking for concurrent events**
- **Machine and device interface dependence and portability**
- **Control timing and scheduling**
  - predictable actions in response to external stimuli
  - deadline (absolute or relative), and jitter
- **Resource constraints and sharing**
  - CPU time, stack, memory, and bandwidth
- **Software abstraction, modular design**
  - information hiding, OO, separate compilation, reusable
  - a sorting procedure -- function, input, output specification